Prefiltered Single Scattering

Oliver Klehm* MPI Informatik
Hans-Peter Seidel† MPI Informatik
Elmar Eisemann‡ Delft University of Technology

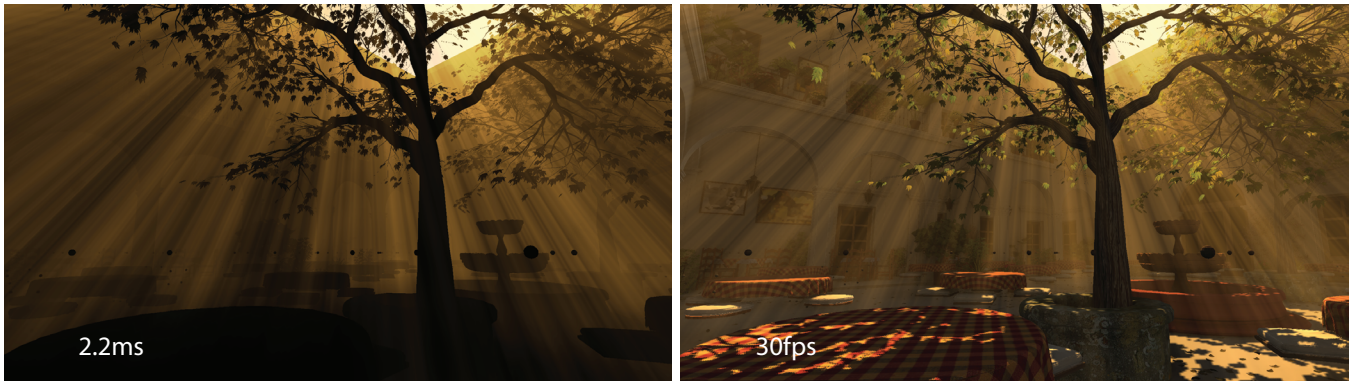

Figure 1: *Our method needs only 2.2ms for single scattering using a* 1024^2 *shadow map and a* 1920×1080 *full-HD screen resolution. It has basically a constant cost per screen pixel, achieving up to a* $20\times$ *speedup and* $6\times$ *less memory consumption for similar quality compared to state-of-the-art methods that struggle with the complexity and details of the scene [Chen et al. 2011]. (Total render time is around 33 ms — most time is spent on ambient occlusion, alpha matting, and disabled backface culling to avoid geometry errors in the scene).*

Abstract

Volumetric light scattering is a complex phenomenon that is difficult to simulate in real time as light can be scattered towards the camera from everywhere in space. By assuming a single-scattering model, we can transform the usually-employed ray-marching into an efficient ray-independent texture filtering process. Our algorithm builds upon a rectified shadow map as input and we propose an efficient rectification scheme, which could be used by other approaches as well. The resulting scattering method is very fast and almost independent of the screen resolution, but it still produces near-reference results. These properties make it a good candidate for performance-critical applications, such as games.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: scattering, participating media, shadow test

1 Introduction

Participating media have an important impact on the appearance of a scene. The resulting effects can add realism and spatial cues to better convey the scene's layout, or serve artistic purposes. A particularly strong visual effect are crepuscular rays (or so-called god rays), which arise from the interaction of light rays with thin participating media. They are often visually pleasing, but rendering them is usually costly. In contrast to surface-light interaction, which is very local, participating media are everywhere, which makes computations challenging [Max 1986]. Recently, several real-time methods have been proposed to approximate light interaction in homogeneous participating media via a single scattering model. Although very efficient when compared to an accurate evaluation, they are still relatively computationally involved when employed in high-performance applications.

In this paper, we present an algorithm with lower complexity. While previous methods perform a ray-marching-like procedure per pixel that can become costly and is scene-dependent in complexity and viewpoint, our solution uses a constant amount of operations per pixel. Hereby, we usually compute high quality scattering much faster (Fig. 1) and measured a speed up of over a magnitude compared to competing solutions [Chen et al. 2011].

Our algorithm is inspired by recent developments in shadow algorithms. Our idea is to transform a standard shadow map into a special basis representation that, combined with a simplified scattering model, allows us to perform prefiltering. To evaluate a view ray, a single dot product is then sufficient. Additionally, we present a linear rectification that aligns epipolar lines, but avoids a pixel-shader scattering used by previous reprojection schemes [Chen et al. 2011]. Precisely, our contributions are:

- An algorithmic efficient and GPU suited single-scattering method using prefiltering;
- A matrix transform for single-step shadow-map rectification;
- Optimization strategies for an efficient GPU execution.

We first discuss related work, before describing the background (Sec. 3), our basic method (Sec. 4), implementation details (Sec. 5) and further extensions (Sec. 6). We then present results (Sec. 7) and conclude the paper (Sec. 8).

2 Related Work

Real-time computer graphics often assumes a vacuum and ignores participating media. Hereby, light interaction is simplified and

*e-mail:oklehm@mpi-inf.mpg.de
†e-mail:hpseidel@mpi-inf.mpg.de
‡e-mail:e.eisemann@tudelft.nl

only occurs on surfaces. Many rendering techniques, such as shadows [Eisemann et al. 2011], often rely on this property, e.g., shadow maps [Williams 1978] only encode the first surface visible from the light source to determine if a surface is illuminated via a simple lookup. Participating media lead to complex phenomena described by the *radiative transport equation* [Chandrasekar 1960]. Details and formal context are given in Sec. 3. When simulating multiple scattering, even recent acceleration schemes are far from real time [Novák et al. 2012], with the exception of diffusion models that support only coarse or no object visibility (e.g., [Billeter et al. 2012]).

Real-time solutions usually rely on single scattering (light is scattered once in the medium towards the camera along the view ray). They are often inspired by image-based strategies, such as deep shadow maps [Lokovic and Veach 2000]. Here, a 1D attenuation function is derived per shadow-map texel, which reduces the light-ray evaluation cost. Our approach renders the light and view-ray evaluation efficient.

Interesting phenomena still occur under the assumption of homogeneous media, such as god rays, which were first simulated by Max [1986]. Although analytical models exist when neglecting visibility [Pegoraro and Parker 2009], stochastic solutions, even when involving efficient ray caches and lookup strategies, remain costly [Sun et al. 2010]. In real-time approaches, the contributions along the view ray can be accumulated by ray marching with an additional shadow test using a simple depth map [Toth and Umenhofer 2009], or by blending parallel planes [Dobashi et al. 2002]. A more efficient traversal was proposed via the rectification of the shadow map (view rays are parallel and have constant depth) [Baran et al. 2010]. This rectification, coupled with a min-max acceleration structure allowed for the reduction of the number of marching steps and gave rise to the algorithm, which is currently considered the real-time reference [Chen et al. 2011].

It is possible to reduce the number of steps further by deriving a marching interval via shadow volumes [Wyman and Ramsey 2008], but such an approach is less useful for complex scenes. A more successful shadow-volume application is their construction from a shadow map instead of the geometry. When rendering the shadow-volume faces with a special shader, an accumulation in the frame-buffer results in a convincing single-scattering approximation [Billeter et al. 2010]. Nonetheless, the performance depends highly on the shadow map resolution and the method requires a high fill rate.

Performance-wise, voxelized shadow volumes are interesting [Wyman 2011]. However, quality and speed depend highly on the chosen voxelization method, which can be costly or is likely to be error-prone. Storing a required high-resolution grid can also lead to an extensive memory consumption. Further, the algorithm factors visibility, which can lead to coarse results and textured light sources can only be handled via full ray marching.

Image-based approaches also have a potential for high performance, but so far, they have been very approximate [Mitchell 2007] or the performance depends strongly on the viewpoint and scene [Engelhardt and Dachsbacher 2010]. The latter approach performs an edge-aware filtering process along the epipolar lines in screen space based on the assumption that significant changes occur at discontinuities. The performance of almost all solutions (e.g., [Wyman 2011] already made use of this algorithm) could potentially be reduced. Nevertheless, the assumption can fail and for detailed scenes, the edge detection can make many samples necessary and might not pay off.

Single scattering methods often have links to shadow algorithms, as does our approach. Following a recent trend [Annen et al. 2007; Annen et al. 2008; Jansen and Bavoil 2010], we represent the visibility test via basis functions. Furthermore, we introduce a new rectification that is compatible with other approaches as well and comes at no additional cost, similar to warping strategies for shadow maps [Stamminger and Drettakis 2002; Wimmer et al. 2004; Martin and Tan 2004].

3 Background

In this section, we give an overview of the used scattering model and the shadow map rectification, which has been widely used in related work [Baran et al. 2010; Chen et al. 2011; Wyman 2011]. Although we will introduce a different rectification as well, it is easier to follow the algorithm using this known transformation.

3.1 Scattering Model

The *radiative transport equation* (RTE) [Chandrasekar 1960] represents physical scattering via a cross-section model; a hypothetical area describing the likelihood of light being absorbed, scattered, or emitted in a specific region of the volume. In its integral form, the RTE defines how much radiance L_i reaches a point in space, e.g., the camera $\mathbf{x}$ from a certain direction ω_i. Assuming no emission, it consists of the following two terms:

$$L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i}) = T_r(\mathbf{x},\mathbf{x}_s)\, L_s(\mathbf{x}_s,-\omega_\mathrm{i}) + \int_0^s T_r(\mathbf{x},\mathbf{x}_t)\, \sigma_\mathrm{t}(\mathbf{x})\, L_\mathrm{scat}(\mathbf{x}_t,\omega_\mathrm{o})\, dt. \quad (1)$$

The first summand describes the amount of radiance that is reflected towards $\mathbf{x}$ from the first encountered surface point $\mathbf{x}_s := \mathbf{x} - s\omega_\mathrm{i}$, where $s \in \mathbb{R}$. This radiance is attenuated by the participating media based on its transmittance: $T_r(\mathbf{x},\mathbf{x}_s)$, which describes the likelihood that a light particle travels through the volume without hitting a volume particle. The second summand in the RTE is the in-scattered radiance at each point $\mathbf{x}_t$ along the view ray from $\mathbf{x}$ to $\mathbf{x}_s$ and again attenuated by the volume, which depends on the amount of particles at position $\mathbf{x}_t$, defined by the extinction coefficient σ_t. Assuming single scattering, it can be expressed as the following sum over all light sources:

$$L_\mathrm{scat}(\mathbf{x},\omega_\mathrm{o}) = \rho\, f \sum_{l=1}^{\text{lights}} V(\mathbf{x},l)\widetilde{L}_\mathrm{i}(\mathbf{x},l)$$

where f is the phase function (the volumetric equivalent to BRDFs for surfaces), which describes how light is scattered in the volume, and ρ is the albedo coefficient. $\widetilde{L}_\mathrm{i}(\mathbf{x},l)$ denotes the incoming radiance from light source l without occlusion and $V(\mathbf{x},l)$ denotes the visibility between the light source and $\mathbf{x}$. Throughout the paper, we consider a single light source as light transport is linear and can be computed per light source by accumulating the results. Further, we make the following assumptions similar to previous real-time work; the phase function is constant and the same everywhere, and the volume homogeneous, i.e., σ_t, ρ are constant. Hereby, we obtain $T_r(\mathbf{x}_a,\mathbf{x}_b) = e^{-||\mathbf{x}_b-\mathbf{x}_a||\sigma_\mathrm{t}}$, making the first term of $L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})$ easy to compute. The real challenge lies in the second term:

$$\rho\, f\, \sigma_\mathrm{t} \int_0^s e^{-t\sigma_\mathrm{t}}\, V(\mathbf{x}_t)\widetilde{L}_\mathrm{i}(\mathbf{x}_t)\, dt. \quad (2)$$

3.2 Rectified Shadow Map and Epipolar Geometry

In a rectified shadow map, view rays are parallel to the x-axis and in turn are parallel to each other (Fig. 2). Consequently, the view rays

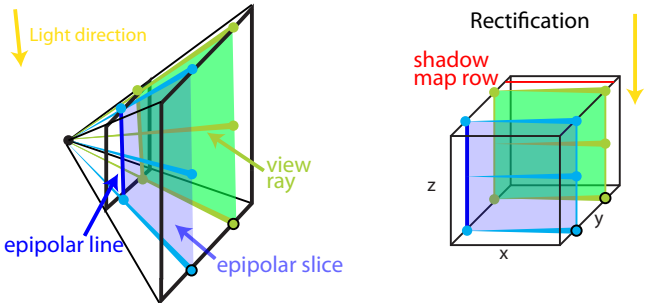

Figure 2: *Rectification makes camera view rays parallel to each other. View rays in the same epipolar slice map to the same y. Finally, the light direction is aligned with the z axis.*

are parameterized by two parameters, y to indicate the shadow map row and its depth z, which is the angle the ray direction spans with the direction of the light source and the camera. View rays go from left to right and have constant y, z, such that they can be mapped to a range $[0; 1]$. All view rays sharing the same shadow-map row y lie in a so-called epipolar slice. In turn, the slice projects on an epipolar line in the camera view. The advantage of such a rectification is that one can accelerate the ray marching along view rays, e.g., by a 1D min-max mipmap [Chen et al. 2011]. The construction of this rectified shadow map is usually done in a pixel shader (details can be found in [Chen et al. 2011]). As blockers between the light and the camera frustum can still produce shadows, they are transferred from the original shadow map and their depth value in the rectified map is clamped to the range $[0; 1]$. In Sec. 6, we will propose an alternative rectification approach.

4 Method

The main idea is to approximate the visibility function for an epipolar slice via a set of basis functions that are independent of the chosen ray. This will enable us to change the formulation of Eq. 2 and extract all ray-related terms from the integral. The remaining expression is an integration of the basis functions, which is performed for entire epipolar slices. Because the visibility function is derived from the discrete shadow map, the integration becomes a prefix summation. The scattering result of a ray can then be obtained via a dot product between the easy-to-compute ray coefficients and the integrated basis functions. Thus, the algorithm has three steps: computing the rectified shadow map, changing basis and computing prefix sums, and evaluating the ray-dependent in-scattering. In the following sections, we present the core of our method, explaining the basis representation and our prefiltering step.

4.1 Scattering

Our goal is to solve Eq. 2 for a given view ray $r := (\mathbf{x}, \omega_\mathrm{i})$. In the rectified shadow map, the view ray r has constant y (epipolar slice) and constant z (depth in shadow map). Due to the discretization of the shadow map, the binary function V becomes piecewise constant and can be defined by functions V_{z_i} that depend on the depth values z_i at texels i stored in the shadow map. Along the ray r, each texel corresponds to a distance Δ in space. Thus, Δ corresponds to the world step size in a ray-marching process. It should be noted that Δ is different for different rays, even in the same epipolar slice. For the moment, we will ignore this issue, Eq. 2 then becomes:

$$L_\mathrm{i}(r) = \rho\, f\, \sigma_\mathrm{t} \sum_{i=1}^{s} V_{z_i} \int_{(i-1)\Delta}^{i\Delta} e^{-t\sigma_\mathrm{t}}\, \widetilde{L}_\mathrm{i}(\mathbf{x}_t)\, dt,$$

s denotes the texel index of the first visible surface $\mathbf{x}_s$, hit by ray r.

Further, to simplify the explanation, we will assume that the light source is directional, i.e., $\widetilde{L}_\mathrm{i}(\mathbf{x}) = \widetilde{L}_\mathrm{i}(\mathbf{x}')\ \forall \mathbf{x}, \mathbf{x}'$. Point lights are discussed in Sec. 6. Our assumption allows us to remove $\widetilde{L}_\mathrm{i}$ from the integral and solve it analytically:

$$L_\mathrm{i}(r) = \rho\, f\, \widetilde{L}_\mathrm{i} \sum_{i=1}^{s} V_{z_i}\, T(i, \Delta), \tag{3}$$

where $T(i, \Delta) := e^{-(i-1)\Delta\sigma_\mathrm{t}}(1 - e^{-\Delta\sigma_\mathrm{t}})$.

Next, we follow the decomposition by Annen et al. [2007], which will allow us to cast the scattering into a prefiltering process. First, we represent each visibility function V_{z_i} by a linear combination of basis functions $B_{k,i} := B_k(z_i)$, such that $V_{z_i}(\mathbf{x}_t) \approx \sum_k^M a_{k,i}(\mathbf{x}_t) B_{k,i}$, with $\mathbf{x}_t$ being the position along r. Key of the linearization is that, first, only the coefficients $a_{k,i}$ depend on the sample points $\mathbf{x}_t$ of r and, second, they only take the z coordinate of $\mathbf{x}_t$ as input, which, because of the rectification, is constant along the entire ray r. Consequently, the coefficients $a_1 \dots a_M$ are independent of i and constant for the entire ray r, and we simply write $a_k(r_z)$. This independence allows us to swap the order of the sums:

$$\begin{aligned} L_\mathrm{i}(r) &\approx \rho\, f\, \widetilde{L}_\mathrm{i} \sum_{i=1}^{s} \left(\sum_{k}^{M} a_k(r_z) B_{k,i}\, T(i, \Delta) \right) \\ &= \rho\, f\, \widetilde{L}_\mathrm{i} \sum_{k}^{M} a_k(r_z) \left(\sum_{i=1}^{s} B_{k,i}\, T(i, \Delta) \right) \end{aligned}$$

In Annen et al.'s approach, the inner sum was completely independent of the ray r and an independent prefiltering became possible. Our situation is different; first, the ray-dependent Δ is in the weighting $T(i, \Delta)$, second, r defines s, the first hit along r and, thus, we need to find a way to adapt the summation appropriately. (Please see the Appendix for an example of basis functions and coefficient computations.)

Weighting Along the Rays To solve the issue regarding Δ, we will assume that we can find a good constant for all rays inside the same epipolar slice. To this extent, we pick a reference ray in the epipolar slice whose Δ (world step size corresponding to a shadow map texel) will be used for all rays. In fact, the Δ of another ray in the same slice deviates by a scale factor c. Hence, we can find an optimal reference ray, for which c is close to one for all rays. It can be shown that the actual factor c is given by the cosine of the angle in epipolar space between the reference and any other view ray. Hence, taking the ray defined by the bisectrix between the two limit view rays in an epipolar slice leads to an optimal Δ_opt.

Adapting the Summation To overcome the second problem of defining s, we compute and store for each basis k $\sum_{i=1}^{s} B_{k,i}\, T(i, \Delta_\mathrm{opt})$ for all possible s. Possible s are directly given by the number of columns of the shadow map. Thus, the resulting *filtered-basis textures* share the resolution of the shadow map, rows still correspond to epipolar slices, but each texel holds the sum of the first s weighted basis function values. For each k the values are stored in the channels of the textures (resp. 2D layers), resulting in a *filtered-basis vector* at each 2D texel position. The filtered-basis textures can be computed efficiently by using a prefix-sum-like solution; in the rectified shadow map, $s = 0$ and $s = s_\mathrm{max}$ map to the left-most and right-most texel of a shadow-map row, respectively. Hence, we perform the sum from left to right for each row (epipolar slice) and weight each $B_{k,i}$ using $T(i, \Delta_\mathrm{opt})$.

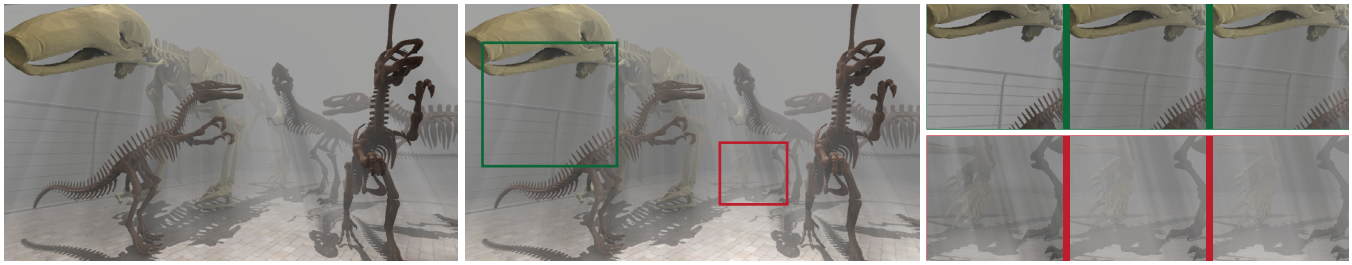

Figure 3: *Left: our method; center: ray-marched in-scattering; The insets show from left to right: our method using unoccluded in-scattering modulated by average visibility; our method applying our weights based on the media transmittance; ray-marching reference with correct attenuation. Average visibility gives plausible results, but many significant differences occur. Shadows are missing or are overly dark. The difference increases with the thickness of the participating media.*

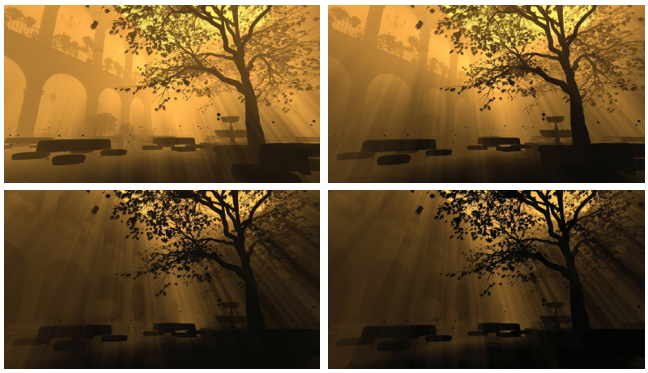

Figure 4: *Alternatives to Fourier series. Top: variance shadow maps (left), exponential variance shadow maps (right). Bottom: our method (left), reference (right). Filterable, efficient statistical methods approximate V too coarsely. Note, we used factored visibility here, resulting in a different result than in Fig. 8.*

It is noteworthy that previous work completely factors out visibility [Wyman 2011] or uses low-rank approximations [Baran et al. 2010; Chen et al. 2011]. Our $T(i, \Delta_{\text{opt}})$ weights are a good approximation with a positive impact on the visual quality (Fig. 3).

Evaluating a Ray Given the filtered-basis textures, the evaluation of scattering along a ray r reduces to a dot product between two vectors and is a constant-time evaluation. The first vector corresponds to the coefficient vector $(a_1, \ldots, a_M)$ that we can compute from r. For the second, we transform the first hit point $\mathbf{x}_s$ along r into the rectified shadow map, and retrieve the filtered-basis vector from the corresponding texel in the filtered-basis textures.

4.2 Fourier Basis

One important choice regarding the algorithm are the basis functions to use, which can have a strong impact on quality and performance. In practice, we found that a Fourier series [Annen et al. 2007] is the best choice for our purposes (see Appendix). We tried alternatives, such as exponential functions [Annen et al. 2008; Salvi 2008]. Unfortunately, they fail because the resulting approximation of V leads to values that exceed one by far when evaluating them off the surface. While this effect is less problematic for surface shadows, where the function is actually evaluated on the surfaces that define V, it is a real problem for scattering because the function is used everywhere in space. Other alternatives include statistical methods, such as variance shadow maps [Donnelly and Lauritzen 2006] and exponential variance shadow maps [Lauritzen and McCool 2008] that give a too coarse approximation (Fig. 4).

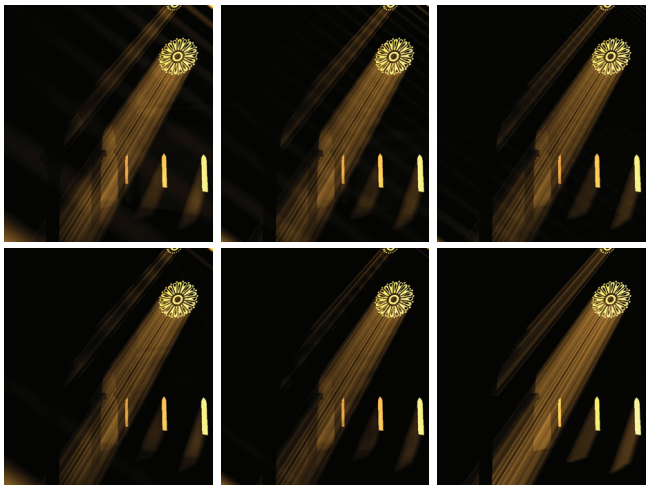

Figure 5: *Difficult scene for ringing. Top row: $M = 8$, $M = 16$, $M = 32$ coefficients (left to right); bottom row: our suppression for $M = 8$, $M = 16$ (left, middle); bottom right: ray-marching reference. The suppression renders the image a bit darker, but disturbing artifacts are reduced.*

When using a truncated Fourier series, the typical approximation artifact is a ringing effect. Annen et al. [2007] propose to apply a scaling and offsetting to V to reduce ringing. Offsetting is used to avoid self shadowing at the surface due to the truncation, which is not problematic in our case. However, light bleeding for fully or mostly occluded rays and shadowing for mostly visible view rays can be disturbing. By using $1.04(V - 0.5) + 0.52$ instead of the visibility function V, we basically shift the ringing artifacts beyond one and below zero. A zero-one clamping needs then to be applied to obtain the final scattering result. The image becomes slightly dimmer, but in practice more pleasing as shown in Fig. 5. Please notice that this is a particularly bad scenario for our method because the background is uniformly colored, making the ringing most visible and the window is small, leading to a very difficult case for the reduction and base representation. The empirically found value 1.04 seems stable and was kept constant for all our test scenes.

5 Implementation

Prefiltering We store the basis functions in a Texture2DArray, which will be transformed into the filtered-basis texture. Interest-

ingly, the simplest implementation of the prefix sum also turned out to be the fastest. We implemented a compute shader that executes one thread per row in the rectified shadow map. Starting on the left and proceeding to the right, the values are written into each texel of the filtered-basis texture. Due to the rectification, this texel order corresponds to a parallel execution over epipolar slices, traversing along the view rays. Hereby, it is simple to maintain and update the weights $T(i, \Delta_{\text{opt}})$. In consequence, only very few registers are needed and the execution is fast.

We also tested a GPU-optimized prefix sum [Harris et al. 2007] designed for sums of complete arrays (while we only require the prefix sum per row and per layer). It makes use of shared memory and avoids some synchronization by performing work in predefined warp sizes. However, this variant is two times slower than the simple approach. Probably, remaining synchronization steps and shared memory create a bottleneck, but this may depend on the used hardware (for us an NVIDIA Titan card).

An interesting observation is that we have the possibility to tradeoff work between the prefiltering stage and the final ray evaluation. Instead of performing the prefix sum for an entire row, we can split the row uniformly, e.g., into three regions, and compute the sums independently, hereby reducing the workload. To evaluate a ray r, we then need several lookups; one from the region that contains the first visible surface along r, and one more for each filtered-basis texture region in front of it — with three regions, maximally three lookups. In practice, three regions led to a speed up of 18%.

A final important observation is that instead of storing a prefix sum of the actual integral, we can store a weighted integral, i.e., we divide the sum of weighted basis functions by the sum of weights $\sum_i^s T(i, \Delta_{\text{opt}})$. Mathematically, it is easy to recover the correct value by multiplying with $\sum_i^s T(i, \Delta_{\text{opt}})$, which can be determined analytically because $\sum_i^s T(i, \Delta_{\text{opt}}) = \sum_i^s \int_{(i-1)\Delta_{\text{opt}}}^{i\Delta_{\text{opt}}} e^{-t\sigma_t} dt = \int_0^{s\Delta_{\text{opt}}} e^{-t\sigma_t} dt$. To evaluate the scattering, we thus perform the dot product as before, but then multiply by the integrated weights up to the first hit point, which are computed analytically. The numerical advantage is that lower precision values can be used to store the filtered-basis textures — no improvement was visible for textures with more than 8bit — leading to a significant reduction in memory and an increase in performance. Further, our ringing reduction is improved. After the dot product the values are generally higher, but still have to lie between zero and one, clamping at this moment, makes the result more accurate.

Shadow-map Resolution The shadow map resolution defines implicitly the number of epipolar slices (y-axis) as well as the number of integration steps (x-axis). The number of needed epipolar slices can be computed from the screen resolution, as well as the configuration of camera and light. Following Baran et al. [2010], we can derive a minimum number of slices such that the distance of a pixel center to the closest epipolar slice is within a predefined bound (e.g., half a pixel).

We use $8\times$ RGBA8 (32 values) for $M = 16$ coefficients (sine and cosine coefficients), which proves sufficient in practice. In theory, compared to the min-max tree of [Chen et al. 2011], we require 2.6 (32bit floats for depth, min-max) / 5.3 (16bit floats) times more memory. Nevertheless, we can rely on hardware filtering when accessing the prefiltered basis functions, countering any potential undersampling artifacts. This possibility allows us to reduce the resolution drastically (usually by a factor of six). Using filtering is beneficial along the x-axis, which averages discrete integration steps, as well as along the y-axis, which blends neighboring epipolar slices. The latter is often desired as it avoids unnaturally sharp edges without requiring an extensive oversampling. Due to hardware filtering our method achieves high performance and quality constantly. Previous work needs higher resolution textures [Chen et al. 2011] or relies on customized upsampling strategies [Baran et al. 2010; Wyman 2011].

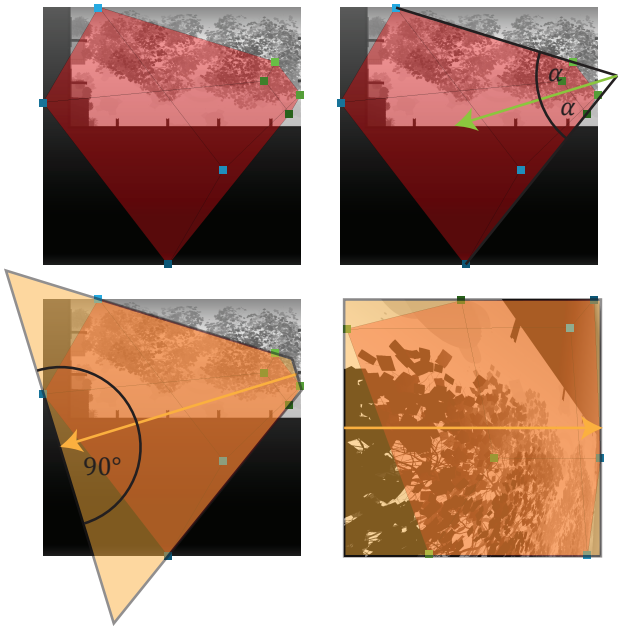

Figure 6: *Rectification (top left to bottom right): original situation; bisectrix; light frustum; rectification*

6 Extensions

In this section, we discuss several extensions to our method; we introduce a novel linear rectification scheme and explain the use of textured and spotlight sources.

6.1 Rectification

As a standard rectification method, we use Baran et al.'s [2010] resampling approach. We render a standard shadow map, find suitable bounds in x, y, z of the rectified light space by analyzing the camera frustum corners and reproject the shadow map into the rectified light space. Although performance is high (1024×1024: 0.1ms, 4096×1536: 0.6ms, 4096×4096: 1.5ms), it is preferable to avoid resampling in general.

Our linear rectification avoids the reprojection completely by constructing a special projection matrix to replace the standard light view-projection matrix when rendering the shadow map. Points can be mapped easily into this space via a simple homogeneous transformation, making lookups very simple as well. To find the transformation, we construct a frustum with an associated matrix that maps the scene in such a way that it fulfills the constraints of our scattering approach, i.e., all view rays in an epipolar slice map to a row of the shadow map, the depth along a view ray is constant, rays in this light space go from left to right, and the shadow map tightly encompasses the camera frustum.

Constructing the Linear Rectification Matrix Fig. 6 gives an overview of the construction steps for a directional light source. We first determine the 3D bisectrix of the camera frustum projected along the light direction (Fig. 6, top-right). The light frustum is

centered on the original camera center. Its up-vector points along the light direction, its view direction along the bisectrix (hereby, near and far plane are parallel to the light direction as well). The frustum's side and top planes are chosen to encompass the camera's near and far plane corners tightly (Fig. 6, bottom-left). Applying the frustum's matrix to the scene leads almost to the wanted result, but, in this light space, the view direction has become the z-axis, and the light direction the y-axis. Hence, to produce the shadow map for the light, we need to swizzle the axes to map the z-axis on the y-axis and vice versa. The final matrix can be used to generate a rectified shadow map with all needed properties (Fig. 6, bottom-right).

The construction is not possible if any of the view rays is parallel to the light direction (as for all such remapping techniques), i.e., when looking directly into the light source. It is easy to detect these cases [Stamminger and Drettakis 2002], and we switch back to a texel-wise reprojection. Additionally, using depth clamping (NV_DEPTH_CLAMP) when rendering the shadow map makes sure that objects shadowing the camera frustum, but lying outside of it, are rendered in the shadow map as well.

Non-linearity of the Rectification Our frustum-based rectification results in a non-linearity along the view rays (x-axis in the shadow map), similar to the non-linearity of the depth buffer. The effect increases the more view and light rays are parallel to each other. Thus, we also return to texel-wise reprojection for close-to-degenerated cases. However, the increased shadow map resolution near the observer can be an advantage, as, due to transmittance, the in-scattered light near the observer receives more precision.

It is possible to reduce the non-linearity similar to split shadow maps [Zhang et al. 2006]. Here, we split the light view into different regions (from left to right), which increase in size. If each region is then rendered into a shadow map of the same resolution, the higher resolution near the observer is counteracted. Although we implemented this solution, it does not lead to a benefit. Nonetheless, it could be an interesting avenue for future research.

Weighting Along the Rays Using the rectification has an impact on the weights $T(i, \Delta)$ that modulate the prefix sum in the scattering approach (cf. Eq. 3). In our rectification, the traveled distance along a view ray increases by a constant factor d from one shadow-map texel to the next. This result stems from the fact that a perspective transformation, such as our rectification, preserves the cross ratio of points [Heckbert 1989]. The weight then becomes $T'(i, \Delta) = e^{-\Delta x \sigma_t \sum_{j=1}^{i} d^{j-1}}(1 - e^{-\Delta x \sigma_t d^i})$. To compute d, we transform the first three texel centers of a shadow-map row into world space (the depth does not matter, but should be the same for all three) and compute the cross ratio of the distances. For a given shadow-map row, the factor d only needs to be determined once before launching its prefix sum.

6.2 Light Source Extensions

In this short subsection, we give a brief overview of extensions that can be used to support a larger variety of light sources, although, we did not implement these solutions.

Textured Sources Integrating textured sources is straightforward; during the prefix sum, we can weight each texel by the light's texture map. In the same way, an angular falloff can be applied. If the light texture is colored, we store one value for each component, instead of a single value.

Shadow Map Resolution	Count Coefficients	Performance in ms Prefix Sum	+ Scattering	= Sum	Memory in MB
1024x1024	16	0.94	1.30	2.24	32
512x1024	16	0.54	1.47	2.01	16
2048x1024	16	2.17	1.31	3.48	64
4096x1024	16	4.32	1.35	5.66	128
1024x512	16	0.97	1.47	2.44	16
1024x768	16	1.03	1.50	2.53	24
1024x1536	16	1.78	1.52	3.31	48
1024x1024	8	0.96	0.92	1.88	16
1024x1024	32	3.07	2.63	5.71	64
1024x1024	64	4.61	5.00	9.61	128

Figure 7: *Computation time and memory consumption of our method with different parameters. As test scene we used* San Miguel *for a screen resolution of* 1920×1080*, where the method achieves near ground truth results with the green highlighted parameters.*

Spotlights Via the reprojection-based rectification [Baran et al. 2010], integration can be performed as in Eq. 3. In our rectification, the scene and camera are transformed additionally via a perspective transformation that sends the source to infinity. A similar solution has been employed in [Wimmer et al. 2004].

Omnidirectional sources can be treated as six independent spotlights, whose contributions are simply added.

Distance Falloff A challenging task is to integrate a distance falloff along light rays, which, in contrast to directional sources, is meaningful for local light sources. This addition is also difficult for other approaches, and we could follow the suggestion of [Baran et al. 2010] to represent the falloff as a set of basis functions. Nonetheless, such a solution is a bit cumbersome and we would need to store all combinations of falloff and shadow basis functions, leading to an increased memory consumption. Instead, a better alternative would be to exchange our basis functions by a more general set [Jansen and Bavoil 2010]. Note that factoring visibility and accounting for the distance falloff in an analytical solution for unoccluded scattering [Pegoraro and Parker 2009] as in [Wyman 2011] is trivially supported by our approach.

7 Results and Discussion

We tested our prototype on an i7 system with an NVIDIA Titan card. The method was implemented in OpenGL using compute shaders. In this section, we present our results and compare to previous work. We also discuss benefits and limitations of our solution.

All results in this section were computed at full-HD screen resolution (1920×1080). Fig. 8 shows a comparison between different methods. We evaluated quality and performance on scenes with differing complexity. Our method performed in all cases below 3 ms, while still resulting in a high quality rendering.

The upper row in each scene set shows the result with optimal quality settings of each method, i.e., the parameters were chosen as suggested by the authors, although, when not leading to a lower quality, we reduced the shadow-map resolution more. Our approach reached the highest performance. For the San Miguel model, our solution was over $20\times$ faster than the min-max-mipmap approach [Chen et al. 2011]. Their low performance is a direct consequence of the complex depth map; the many leaves of the tree lead to an overhead on the min-max hierarchy (as also observed in [Baboud et al. 2011]). For simple scenes, such as the terrain, the min-max structure reaches optimal performance and our method is only 1.8 times faster (due to lower resolution shadow maps, but hardware filtering). These comparisons show the main benefit of our method: it is independent of the scene complexity and con-

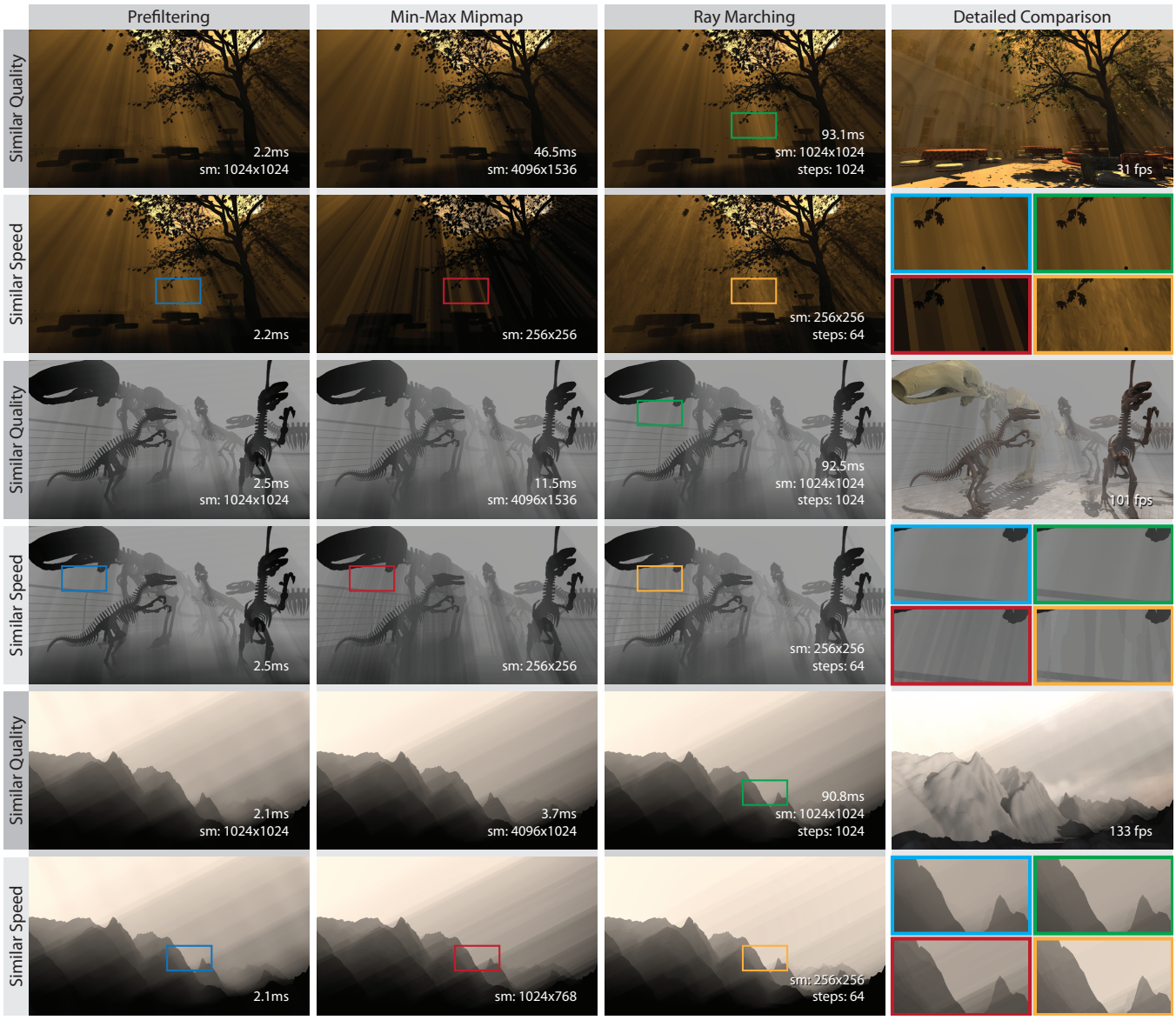

Figure 8: *Quality/Performance comparison at* 1920×1080*. To compare quality, we chose the suggested settings of all methods. For same time, we reduced the shadow map resolution. Our result compares favorably to reference quality, even for low resolution shadow maps.*

figuration, while the cost of other methods is less predictable. In this regard, our solution is very different from other approaches whose run time is more strongly influenced [Billeter et al. 2010; Engelhardt and Dachsbacher 2010; Baran et al. 2010; Billeter et al. 2012]. Precisely, our cost is $\mathcal{O}(wh + ad)$ for a screen resolution of $w \times h$, a maximum of d view-ray integration steps, and the number of epipolar slices a. Brute-force ray marching requires d integration steps per pixel, yielding $\mathcal{O}(whd)$. Acceleration structures [Baran et al. 2010; Chen et al. 2011] can lead to an improvement; potentially $\mathcal{O}(wh \log d + ad)$. However, $\log d$ is the optimal case, which becomes d in the worst case. Our method does not suffer from this scene dependence.

The lower row in each scene set shows an equal time comparison. Here, we adapted the shadow map resolution of the competitors until our performance was roughly matched. The resulting images show significant artifacts stemming from discretization issues. While our method allows for hardware filtering, other approaches do not have this option. Our images remain visually pleasing, even with lower resolution shadow maps. While ringing is a potential issue, our suppression works rather well and scene details (textures, materials, surface lighting…) tend to hide this artifact completely.

A performance breakdown of our solution for San Miguel is shown in Fig. 7. The scattering cost is indeed constant, while the prefix sum scales linearly with resolution, as expected. Changing the amount of coefficients behaves sub-linearly. This result is probably linked to the cache and texture mechanisms; as 32bit floating point computations are standard and on newer cards even 64, it is likely that the bandwidth has been increased correspondingly. Hence, our solution also seems well suited for future hardware developments.

8 Conclusion

We present a very efficient single scattering approach, that only needs constant time for each screen pixel. The prefiltering process

is fast and because our method allows for hardware interpolation, much smaller resolutions can be used for the shadow maps than in previous approaches, hereby we also keep the memory requirements low. In contrast to other solutions, our method delivers predictable, generally high performance and convincing image quality, independent of the scene size, complexity, or detail level. Hence, we believe that it is a good candidate for time-critical applications.

In the future, we will investigate non-homogeneous media. Further, our rectification might motivate rethinking the warping procedure and coupling it closely to the actual ray marching.

Acknowledgements San Miguel was made by Guillermo M. Leal Llaguno. This work was partly supported by the Intel Visual Computing Institute at Saarland University.

References

ANNEN, T., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2007. Convolution shadow maps. In *EGSR*, 51–60.

ANNEN, T., MERTENS, T., SEIDEL, H.-P., FLERACKERS, E., AND KAUTZ, J. 2008. Exponential shadow maps. In *Graph. Interf.*, 155–161.

BABOUD, L., EISEMANN, E., AND SEIDEL, H.-P. 2011. Precomputed safety shapes for efficient and accurate height-field rendering. *TVCG 99*.

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 (Dec.), 178:1–178:10.

BILLETER, M., SINTORN, E., AND ASSARSSON, U. 2010. Real time volumetric shadows using polygonal light volumes. In *HPG*, 39–45.

BILLETER, M., SINTORN, E., AND ASSARSSON, U. 2012. Real-time multiple scattering using light propagation volumes. In *i3D*, 119–126.

CHANDRASEKAR, S. 1960. *Radiative Transfer*.

CHEN, J., BARAN, I., DURAND, F., AND JAROSZ, W. 2011. Real-time volumetric shadows using 1d min-max mipmaps. In *i3D*, 39–46.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2002. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Graph. hardware*, 99–107.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *i3D*, 161–165.

EISEMANN, E., SCHWARZ, M., ASSARSSON, U., AND WIMMER, M. 2011. *Real-Time Shadows*. AK Peters (CRC Press).

ENGELHARDT, T., AND DACHSBACHER, C. 2010. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *i3D*, 119–125.

HARRIS, M., SENGUPTA, S., AND OWENS, J. D. 2007. *GPU Gems 3*. ch. 39. Parallel Prefix Sum (Scan) with CUDA, 851–876.

HECKBERT, P. S. 1989. Fundamentals of texture mapping and image warping. Tech. rep.

JANSEN, J., AND BAVOIL, L. 2010. Fourier opacity mapping. In *i3D*, 165–172.

LAURITZEN, A., AND MCCOOL, M. 2008. Layered variance shadow maps. In *Graph. Interf.*, 139–146.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *SIGGRAPH*, 385–392.

MARTIN, T., AND TAN, T.-S. 2004. Anti-aliasing and continuity with trapezoidal shadow maps. In *EGSR*, 153–160.

MAX, N. L. 1986. Atmospheric illumination and shadows. *SIGGRAPH Comput. Graph. 20*, 4 (Aug.), 117–124.

MITCHELL, K. 2007. *GPU Gems 3*. ch. 13. Volumetric Light Scattering as a Post-Process, 275–285.

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph. 31*, 4, 60:1–60:11.

PEGORARO, V., AND PARKER, S. G. 2009. An Analytical Solution to Single Scattering in Homogeneous Participating Media. *CGF (EG) 28*, 2, 329–335.

SALVI, M. 2008. Rendering filtered shadows with exponential shadow maps. Feb., 257–274.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. *ACM Trans. Graph. 21*, 3 (July), 557–562.

SUN, X., ZHOU, K., LIN, S., AND GUO, B. 2010. Line space gathering for single scattering in large scenes. *ACM Trans. Graph. 29*, 4 (July), 54:1–54:8.

TOTH, B., AND UMENHOFER, T. 2009. Real-time volumetric lighting in participating media. In *EG Short Papers*.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph. 12*, 3 (Aug.), 270–274.

WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. 2004. Light space perspective shadow maps. In *EGSR*, 143–151.

WYMAN, C., AND RAMSEY, S. 2008. Interactive volumetric shadows in participating media with single-scattering. In *IEEE Interactive Ray Tracing*, 87–92.

WYMAN, C. 2011. Voxelized shadow volumes. In *HPG*, 33–40.

ZHANG, F., SUN, H., XU, L., AND LUN, L. K. 2006. Parallel-split shadow maps for large-scale virtual environments. In *Virtual reality continuum and its applications*, 311–318.

A Linearized Visibility via Fourier Series

We linearize the evaluation of visibility by representing the visibility function V via a linear combination of basis functions. As V is piecewise via a set of visibility functions V_{z_i}, we focus on this particular approximation. Precisely, we want to obtain: $V_{z_i}(\mathbf{x}_t) \approx \sum_k^M a_k(z) B_k(z_i)$ with i being the texel index in the rectified shadow map, z_i the corresponding depth, and z the depth in light space of position $\mathbf{x}_t$.

Following Annen et al. [2007], the Fourier series representation then becomes:

$$a_{(2k-1)}(z) = 2c_k^{-1}\cos(c_k z), \qquad a_{(2k)}(z) = -2c_k^{-1}\sin(c_k z)$$
$$B_{(2k-1)}(z_i) = \sin(c_k z_i), \qquad B_{(2k)}(z_i) = \cos(c_k z_i)$$

with $c_k = \pi(2k-1)$. Because the step function is not considered between zero and one, but [-0.5,0.5], we need to keep in mind to apply a constant offset of 0.5 after evaluating the function via the dot product.