Real-Time Screen-Space Scattering in Homogeneous Environments

Oskar Elek Tobias Ritschel Hans-Peter Seidel
MPI Informatik

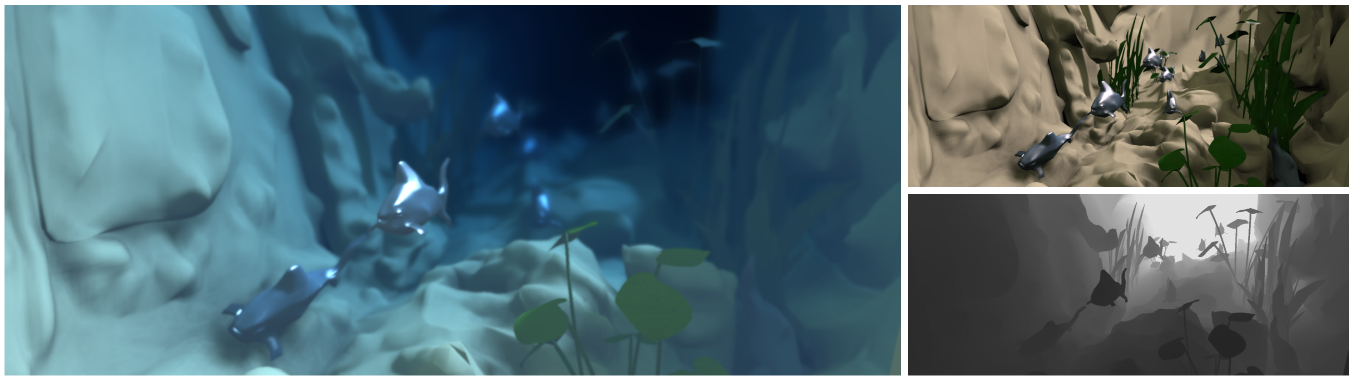

Figure 1: *We reproduce the blurring and colour shifts in participating media such as sea water from a single HDR image and its depth buffer* (right) *in real-time (4.3 ms for the scattering, 40.2 ms framebuffer generation (HDR, PCF soft shadows, SSAO), 2048×1024 resolution).*

Abstract

This work presents an approximate algorithm for computing light scattering within homogeneous participating environments in screen space. Instead of simulating the full global illumination in participating media we model the scattering process by a physically-based point spread function. To do this efficiently we apply the point spread function by performing a discrete hierarchical convolution in a texture MIP map. We solve the main problem of this approach, illumination leaking, by designing a custom anisotropic incremental filter. Our solution is fully parallel, runs in hundreds of frames-per-second for usual screen resolutions and is directly applicable in most existing 2D or 3D rendering architectures.

Keywords: participating media, light scattering, screen-space methods, real-time rendering

1 Introduction

The visual richness we observe around us is a consequence of various different optical phenomena. Among these, light scattering is of large importance, as participating media causing this phenomenon are very common in our environment. Virtually all non-metallic substances scatter light to a certain extent, including such basic compounds as air, water, milk or organic tissues.

Light scattering influences appearance of objects on both local and global scale. Since light can interact with the medium at any point within, simulation of these effects has been challenging for computer graphics algorithms, which were originally based on the simplifying assumption of light interacting only with discrete boundaries of objects which are separated by vacuum.

The local interaction of light with media can to some extent be simulated by sub-surface scattering models. An example of these is the successful method of Jensen et al. [2001] built on the dipole diffusion approximation, which serves as a basis for many approaches even today. In certain cases of very dense objects it is even possible to use a *bi-directional reflectance distribution function* (BRDF) to approximate the light scattered from underneath the surface of a simulated object. An example is the Lambertian reflectance model.

On the other hand, simulating global illumination in the presence of a participating medium is arguably more challenging. Imagine a room filled with smoke; the smoke will influence light propagation in the entire room and therefore the algorithm that attempts to include this effect will not be able to localize it to surfaces as in the previous case. Although most standard global illumination solvers can be extended to include participating media, this is often computationally very demanding and can render a particular method too slow for practical utilization. Moreover, this issue will naturally stand out even more if one is interested in simulating global illumination interactively.

In this work we attempt to design an approximate interactive method to simulate global effects caused by light scattering in *screen space*. Screen-space methods are popular in interactive environments for their usual simplicity and speed. Despite the intrinsic drawback that they neglect parts of the scene that are not visible to the camera they are still very convenient in certain conditions.

Our main intention is to simulate scenes containing a homogeneous participating medium. Prime examples of such scenes are foggy outdoor or underwater environments. As light scattering manifests itself mainly by blurring of the illumination, the central idea of our approach is to use a physically-plausible two-dimensional filter (a *point spread function* [Premože et al. 2004]) to blur the illumination of the rendered scene depending on each pixel's distance. This approach is well known in depth-of-field simulation research, where it has first been presented by Rokita et al. [1993] and frequently used since. To avoid convolution by large spatially varying filter kernels, we adapt and improve a variant of hierarchical anisotropic filtering [Lee et al. 2009]. The main improvement (described in Sec. 3) lies in our modification of the anisotropic Gaussian filter that prevents the *illumination leaking* more effectively.

In the following section we provide some background theoretical information about light scattering and an intuition behind the empirical behaviour of this process.

2 Background

Light scattering is a complex physical phenomenon and under general conditions it is indeed difficult to simulate. On the other hand, understanding the empirical high-level behaviour of this process is much easier. We will therefore first explain the intuition behind this behaviour and then provide a short formal description of the corresponding physics in the second part of this section.

If a beam of collimated light enters a medium some of its photons will scatter and their propagation direction will change from the original beam direction. This will result in blurring of the beam. One can naturally expect that the deeper into the medium the beam progresses, the more photons will scatter out of it. This will provide them more space to spread, causing the beam to get more blurry. The blurring will as well increase with higher *optical thickness* of the medium, simply because the scattering interactions will take place more frequently.

In screen-space methods, instead of considering individual light sources in 3D space explicitly, *every pixel* of the rendered image implicitly is a light source. Based on the description from the previous paragraph we can expect the blurring of every pixel to be proportional to its distance from the camera and on the optical thickness of the medium itself. Accounting for the scattering of the visible portion of the scene then roughly entails:

1. obtaining the correct distance-dependent blurring kernel and
2. convolving the rendered scene image with this kernel.

At this point we would like to formalize this informal description by first reviewing the *radiance transfer equation* (RTE) and then transitioning to a different approach based on a *point spread function* (PSF) formulation, which is more suitable for our purpose.

Radiance transfer equation The RTE describes propagation of light through a participating medium:

$$\frac{dL(\mathbf{x},\omega)}{d\mathbf{x}} = -\sigma_t(\mathbf{x})L(\mathbf{x},\omega) + \sigma_s(\mathbf{x}) \int_{\Omega_{4\pi}} L(\mathbf{x},\omega')p(\omega',\omega)\, d\omega' \tag{1}$$

Eq. 1 describes a differential change of radiance L at the point in space $\mathbf{x}$ in the direction ω. The *extinction coefficient* σ_t is obtained as $\sigma_t = \sigma_a + \sigma_s$, where σ_a is the *absorption coefficient* and σ_s the *scattering coefficient* of the simulated medium. These quantities are wavelength-dependent, which will be implicit throughout the rest of the paper. The spherical function p is called the medium phase function (in homogeneous media p is spatially invariant). It defines an angular distribution of light after it scatters on a medium particle. See Fig. 2 for a depiction of the concept behind RTE.

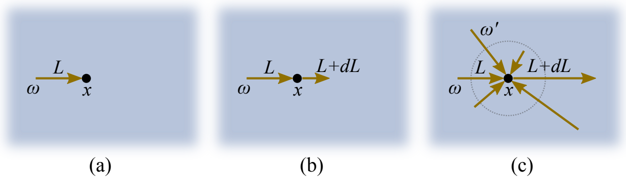

Figure 2: *An illustration of the concept of the RTE: the radiance L at* $\mathbf{x}$ *in the direction* ω (a) *is attenuated by extinction in the medium* (b) *and subsequently increased by integrating radiance contributions from the entire sphere that scatter into* ω (c). *The cases* (b) *and* (c) *correspond to the two terms on the right side of Eq. 1.*

The RTE is a versatile mathematical tool easily extensible to more general types of scattering. One of the main features of this formulation is the locality of its evaluation, which makes it particularly useful in local *finite element* methods. However, in our approach we are looking for a global description of the scattering process, as we are interested in expressing the spreading of light when it propagates from one point in space to another. For this we will make use of the concept of point spread function.

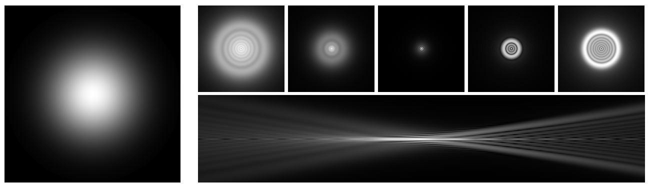

Figure 3: *Two examples of point spread functions: circular Gaussian with standard deviation of 3* (left) *and a circular aberration of an optical system in different levels of de-focusation* (top right) *with the corresponding longitudinal section* (bottom right).

Point spread function A PSF is a function $f(\mathbf{x}) \in \mathbb{R}^2 \to \mathbb{R}^+$ that defines the spreading of a point signal, which is usually represented by the Dirac pulse. Fig. 3 shows an example of a point spread function. PSFs usually cause blurring of the original signal and appear for various causes, e. g., flaws or limitations of measurement systems, or, as in our case, as a direct result of a physical process.

As these blurring effects appear frequently PSFs have been widely adopted in both image processing and computer graphics. For instance, in the context of light scattering, Narasimhan and Nayar [2003] derived a physically-plausible PSF for analysing atmospheric weather conditions, Premože et al. [2003; 2004] applied PSFs to derive a global illumination solver for participating media, and Lopez-Moreno et al. [2008] used this framework for artistically-driven image modification to convey presence of participating media.

A PSF is applied by convolution of the original signal with a particular expression of the PSF, which can be discrete and tabulated or might have a closed analytical form. Based on the explanation we have provided earlier the usefulness of formulating light scattering by a PSF becomes apparent now. Having an expression for the amount of blurring caused by scattering between two points in space, obtaining a version of the rendered image with the scattering accounted for is basically a matter of performing a discrete convolution of the image and the PSF. Naturally, it is also necessary to know the distance to each pixel in the image to locally control the degree of blurring.

Please note that in our case the PSF varies spatially, and hence does not conform to the original definition. We therefore call the functional that maps every spatial location to the PSF that accords to the camera distance of that location $F(\mathbf{x},\mathbf{y}) \in (\mathbb{R}^2 \times \mathbb{R}^2) \to \mathbb{R}^+$. Applying this spatially-varying, distance-dependent PSF to an image is not a simple convolution with the PSF itself, but a convolution with a kernel that is only similar to the PSF. Additionally this kernel will likely be anisotropic, although the original PSF might have been isotropic.

An expression for a spatially-varying scattering PSF has been derived by Premože et al. [2004] based on the *path integral* formulation originally devised for application in quantum physics. The authors assume a Gaussian spatial distribution of the photons scattered from an incident collimated pencil of light. This is a reasonable assumption according to the central limit theorem, since scattering can be regarded as a stochastic process with many realizations. Premože et al. estimate the standard deviation W of the resulting Gaussian

Sidebar 1: Path integral formulation

The radiance transfer equation (Eq. 1) defines the local behaviour of light in the presence of a participating medium. It is however not immediately clear how this behaviour relates to light propagation on a larger scale.

One way to bridge these two paradigms is to use the path integral formulation [Tessendorf 1987; Premoze et al. 2003; Premože et al. 2004; Tessendorf 2011]. According to it the RTE can be reformulated using the so-called Green propagator (evolution operator) G as

$$L(\mathbf{x},\omega) = \int G(\mathbf{x},\mathbf{x}',\omega,\omega')L_0(\mathbf{x}',\omega')\; d\mathbf{x}'d\omega',$$

L_0 being the initial radiance distribution in space. G therefore describes how the initial energy distribution changes when the boundary conditions are taken into account.

It is however not immediately obvious how this mathematical formulation relates to the underlying physical process. What G describes is an averaged high-level behaviour of many individual photons, which are randomly scattered in the medium. As previously noted [Premoze et al. 2003; Premože et al. 2004] a similar process has been described in the field of quantum mechanics by the means of the *sum over particle histories* [Feynman and Hibbs 1965]. This method states that the probability of a particle being in a certain position in space is obtained by superposing all possible paths the particle might have taken within the given boundary conditions.

As Premože and Tessendorf observed, in light transport we can similarly obtain the light intensity in a certain point and direction by integrating all possible photon paths that conform to this boundary condition. This can informally be written as

$$L \sim \int_{\text{all paths}} e^{-\tau(\text{path})},$$

where τ defines an *effective attenuation* of a path (essentially stating its contribution to the result in accordance with the Beer-Lambert-Bouguer law). Obviously the difficult part of this relatively simple idea is to rigorously express the infinitely-dimensional integral over the path space, and this has been the focus of the above-cited works.

Path Integral References

FEYNMAN, R. P., AND HIBBS, A. R. 1965. *Quantum mechanics and path integrals.* McGraw-Hill.

TESSENDORF, J. 1987. Radiative transfer as a sum over paths. *Phys. Rev. A 35*, 872–878.

TESSENDORF, J. 2011. Angular smoothing and spatial diffusion from the Feynman path integral representation of radiative transfer. *J. Quantitative Spectroscopy & Radiative Transfer 112*, 751–760.

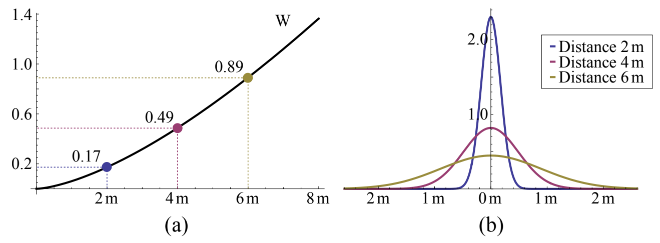

Figure 4: *Behaviour of the function $W(s)$ (Eq. 2) for $\sigma_a = 0.1$, $\sigma_s = 0.3$ and $g = 0.9$: plot of the function in dependence on the travel distance s (in meters)* (a) *and the resulting Gaussian PSF shape for three different travel distances* (b). *The corresponding standard deviations of the curves in* (b) *are marked in* (a).

PSF based on the travel distance s as

$$W(s) = \sqrt{\frac{1}{2}\left(\frac{2\sigma_a}{3s} + \frac{4}{s^3\sigma_s(1-g)}\right)^{-1}} \qquad (2)$$

The scattering asymmetry factor g is defined as an average cosine on the scattering angle θ resulting from the used phase function p as $g = \int_{\Omega_{4\pi}} p(\theta)\cos\theta\, d\omega'$; it is often used as a shape control parameter for analytical phase function expressions, such as the frequently used Henyey-Greenstein function. The behaviour of Eq. 2 is shown in Fig. 4.

We employ this expression in our method as well. For further details about the relation between radiance transport theory and path integral formulation please refer to Sidebar 1. In the next section we present an overview of our approach and then discuss its details in the following sub-sections.

3 Our approach

Having expressed the ideas behind our approach we can now proceed to its algorithmic description. We first provide an overall view of the whole process and then proceed to explain the individual steps in more detail.

3.1 Overview

Input to the algorithm is an HDR image $\mathsf{I} : \mathbb{N}^2 \to \mathbb{R}^3$ containing incoming radiance and a distance map $\mathsf{D} : \mathbb{N}^2 \to \mathbb{R}$ of the same resolution. D contains the world-space distance of every pixel to the camera; the common linear z-buffer is approximately such a distance map. The output is an image J which includes scattering and absorption caused by a specified homogeneous medium.

The challenge of rendering participating media efficiently in screen space is to avoid performing convolutions by large and spatially varying PSF kernels. Instead, our approach approximates scattering computation by a *gathering* computation and performs only constant time local image modifications, combined with a specially constructed MIP map. The algorithm proceeds in three steps (cf. Fig. 5), each of which operates fully parallel over image pixels:

1. First, the attenuation of the medium is accounted for. This step produces three images: $\mathsf{I}' : \mathbb{N}^2 \to \mathbb{R}^3$ which contains the radiance that did not interact with the medium at all, $\mathsf{I}^{[0]} : \mathbb{N}^2 \to \mathbb{R}^3$ containing the radiance that was scattered in the medium, but not absorbed, and $\mathsf{W}^{[0]} : \mathbb{N}^2 \to \mathbb{R}$ which stores additional distance-dependent information necessary to build the MIP map in the next step. Sec. 3.2 describes this step in detail.
2. The second step produces a K-level MIP map $\mathsf{I}^{[1..K]}$ out of $\mathsf{I}^{[0]}$. This MIP map is used to approximate the spatially-varying filter required in Step 3. Every level of $\mathsf{I}^{[1..K]}$ contains an increasingly

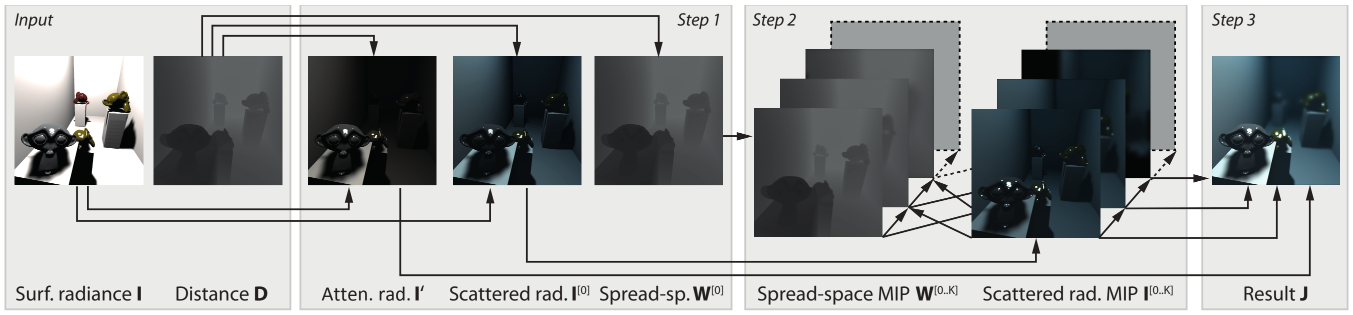

Figure 5: *The three stages of the proposed algorithm shown for a box filled with water and shiny objects* (from left to right)*: The input is first split into attenuated and scattered radiance as well as spread-space distance. Second, the scattering and spread-space MIP maps are constructed (the first three levels are shown). Finally, the result is produced by combining the attenuated and the scattered radiance blurred according to the spread-space distance.*

blurred version of $\mathsf{I}^{[0]}$. Different from classic MIP maps, we account for the non-linearity caused by applying the scattering PSF by a Gaussian gather function. To this end, a second MIP map $\mathsf{W}^{[0..K]}$ storing the so-called *spread-space* distance is constructed and used. This is the core step of the algorithm and is detailed in Sec. 3.3.

3. Finally, the distance s of every pixel in I stored in D is used to select a MIP level to fetch an approximation of the convolution of I with a Gaussian similar to $W(s)$. This fetched value is then simply added to I', producing J. Sec. 3.4 describes this step.

To simplify the notation we will be applying basic arithmetic operations and functions to whole images. The result of such an operation is again an image, where the respective operation is applied per-pixel. Further, we will be using discretized versions of continuous functions, for instance W is a discretized version of W.

3.2 Preprocessing

This step produces the images I', $\mathsf{I}^{[0]}$ and $\mathsf{W}^{[0]}$:

$$\mathsf{I}' = e^{-\sigma_t \mathsf{D}} \cdot \mathsf{I} \tag{3}$$

$$\mathsf{I}^{[0]} = e^{-\sigma_a \mathsf{D}} \cdot (1 - e^{-\sigma_s \mathsf{D}}) \cdot \mathsf{I} \tag{4}$$

$$\mathsf{W}^{[0]} = W(\mathsf{D}) \tag{5}$$

Eq. 3 calculates the attenuated radiance according to the Beer-Lambert-Bouguer law, that is, the fraction of light that reaches the camera directly without interacting with the medium. This is the original colour of the scene, but attenuated according to the distance by σ_t, as seen in Fig. 5. I' will be utilized only in the final Step 3 (Sec. 3.4).

Eq. 4 obtains the part of the radiant energy that *was not* absorbed (first factor) but *was* scattered (second factor) on its way to the camera. This is the energy that will be blurred in Step 2, therefore will show colour shifts caused by the medium absorption, e. g., in water will turn increasingly blue for more distant pixels. Again this effect can be seen in Fig. 5. The consequences of decoupling absorption and scattering in this way are discussed in Sec. 5.1.

Finally, Eq. 5 projects the pixels' distances into what we call the *spread space*. The motivation for this is to design our anisotropic filter so that it more closely matches the scattering behaviour. This will be detailed in the following section.

3.3 Filtering

This subsection introduces the screen-space reference solution, as well as a naïve and our approximation to it.

Screen-space reference To compute scattering in screen space for every pixel location $\mathbf{x}$ in $\mathsf{I}^{[0]}$, one could (quadratically) iterate over all other pixels $\mathbf{y}$ and evaluate how much the (spatially varying) PSF of $\mathbf{y}$ contributes to $\mathbf{x}$ according to $F[\mathbf{y}](\mathbf{y}-\mathbf{x})$. Even if the range of pixels to visit for every pixel can be limited to a smaller neighbourhood—which is generally not possible for HDR conditions or optically thick media—the effort is substantial and can take up to several seconds for a single image. We call this method the *screen-space reference* and compare it to our approach in Figs. 7 and 8.

Naïve approximation If the PSF was spatially invariant, we could compute the convolution of the image with the PSF by building a simple Gaussian MIP map. Gaussian MIP maps can be constructed efficiently, and the convolution of the image and the PSF could be evaluated precisely by a single fetch from the corresponding MIP level, at each pixel.

Using this approach to apply a spatially variant PSF naturally bears some limitations. First, fetching a value for a pixel corresponding to its distance assumes that each pixel from its neighbourhood contributes to it with a PSF of the same size. In addition, the convolution kernel is not even a Gaussian anymore, but the sake of our approximation we assume it can be reasonably approximated by one. We call this the *naïve* method and compare it to our approach in Fig. 6.

This approximation works reasonably well if the PSF does not change abruptly across the image. Unfortunately this is not the case for distance discontinuities on object boundaries. The most visible consequence of such variation is *illumination leaking* (Fig. 6a), which happens because bright image locations close to the camera are blurred into the higher MIP levels that correspond to more distant parts of the scene. When the radiance of a distant pixel is then read from the corresponding MIP level, it is influenced by this leaked radiance, although it should not be as the source highlight is closer to the camera. Regrettably, this issue is especially apparent for scenes where the effect we try to render is most desired: scenes with high dynamic range, such as the one shown.

Our approximation We remedy this problem in a similar manner as done for depth-of-field rendering by Lee et al. [2009]. To prevent leaking of illumination from "closer" MIP levels to higher, more "distant" levels during the MIP map construction we need to know

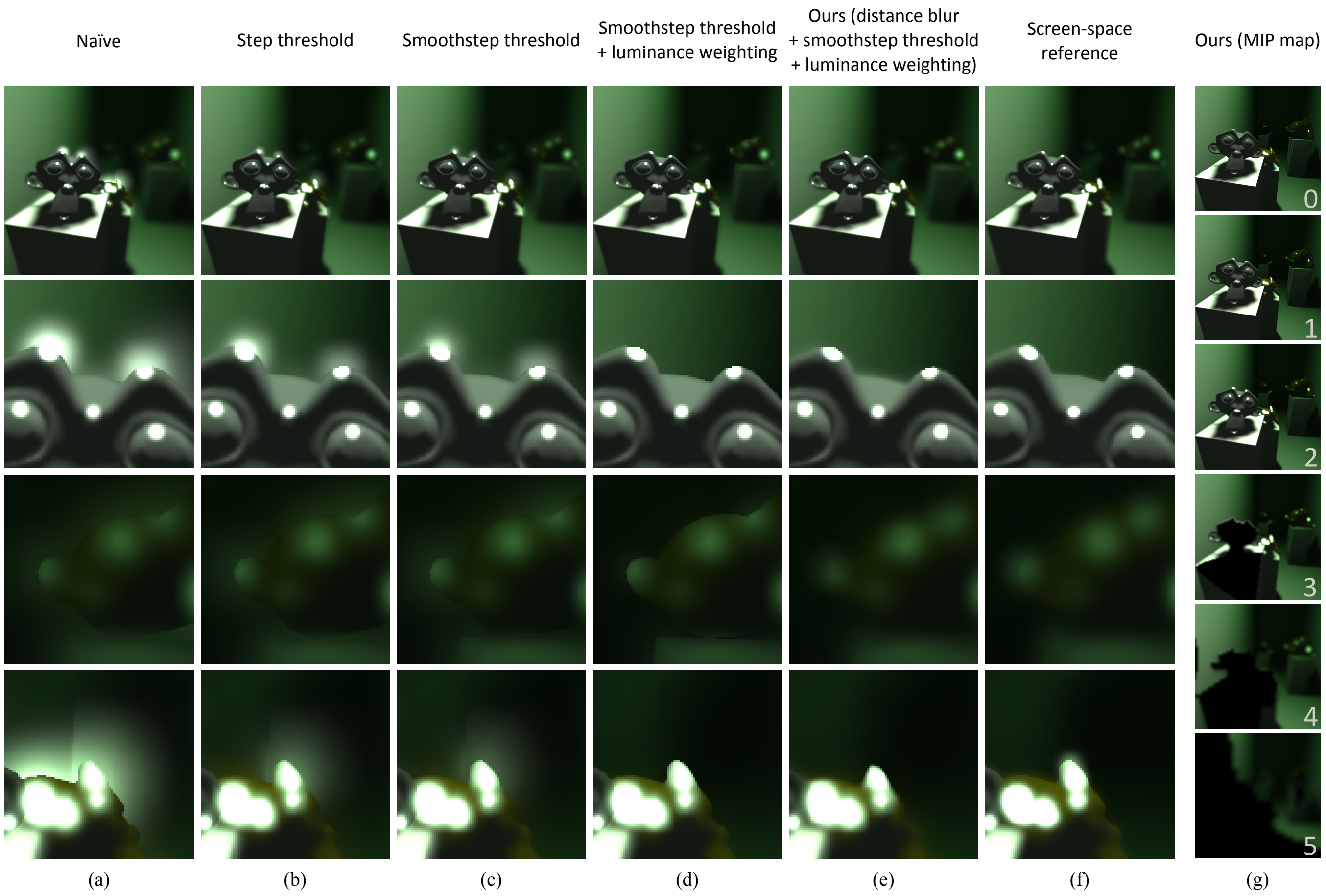
Naïve
Step threshold
Smoothstep threshold
Smoothstep threshold + luminance weighting
Ours (distance blur + smoothstep threshold + luminance weighting)
Screen-space reference
Ours (MIP map)
0
1
2
3
4
5
(a)
(b)
(c)
(d)
(e)
(f)
(g)

Figure 6: *Comparison of our method in various stages and with other methods. Applying the standard Gaussian MIP map* (a) *leads to light leaking, which is slightly reduced by applying step* (b) *and the smoothstep* (c) *thresholding, but then significantly by weighting the spread-space distance by the pixels' luminance during the averaging* (d)*. The discontinuities that still remain are successfully removed by blurring the distance map* (e)*. The screen-space reference* (f) *resulting from a spatially variant Gaussian PSF computed by gathering from a 100×100-neighbourhood is three orders of magnitude slower (2.1 s vs. 2.7 ms for our solution).* (g) *shows the scattering MIP map used to produce* (e)*.*

the distance of the filtered pixels and incorporate it into the filtering process. The resulting *anisotropic* filter then masks out such leaking pixels as the higher MIP levels are constructed. For this we construct $\mathsf{W}^{[0..K]}$ containing the distance projected to spread-space, which can be used in the same manner as the distance itself thanks to the direct dependence between them (cf. Fig. 4a). The spread-space distance performs better for our purpose, because it allows us to directly relate the MIP levels with the distance-dependent masking thresholds, as described in a moment.

We first describe the construction of the MIP pyramid $\mathsf{I}^{[1..K]}$. To obtain $\mathsf{I}^{[k]}$ (where $k = 1..K$) we need to mask out those pixels of $\mathsf{I}^{[k-1]}$ during the filtering that correspond to smaller distances in the scene, in order to avoid the light leaking:

$$\mathsf{I}^{[k]} = (\mathsf{M}^{[k]} \cdot \mathsf{I}^{[k-1]}) * \mathsf{G}', \tag{6}$$

where G' is a simple discrete Gaussian-like filter, in our case a 4×4 filter with weights $\{0.13, 0.37, 0.37, 0.13\}$ for each dimension. As shown by Burt [1981], for the given size this is the filter that leads to the smallest deviation from the exact Gaussian distribution. The auxiliary mask M at level k can be defined as

$$\mathsf{M}^{[k]} = \mathrm{smoothstep}(T, (1+\varepsilon)\cdot T, \mathsf{W}^{[k-1]}). \tag{7}$$

The definition uses the standard smoothstep function to perform the masking smoothly. The parameters T and ε control the masking threshold distance and width. As shown by Burt [1981], when a discrete hierarchical convolution as in our case is performed, the width of the corresponding result filter doubles with every level of the hierarchy, and therefore the threshold that separates these levels must double in size as well. Because of this we define $T = c \cdot 2^{k-1}$, where c is a scaling constant described in Sec. 3.4.

Simultaneously with $\mathsf{I}^{[1..K]}$ we build the MIP chain $\mathsf{W}^{[1..K]}$. The MIP map $\mathsf{W}^{[0..K]}$ holds average spread-space distances for whole groups of pixels at the corresponding levels, as they are needed for computation of $\mathsf{I}^{[1..K]}$ (Eq. 7):

$$\mathsf{W}^{[k]} = \frac{(\mathsf{Y}^{[k]} \cdot \mathsf{W}^{[k-1]}) * \mathsf{U}}{\mathsf{Y}^{[k]} * \mathsf{U}}, \tag{8}$$

where U is the uniform distribution of the same size as G'. We further define the auxiliary mask Y at level k as

$$\mathsf{Y}^{[k]} = y(\mathsf{I}^{[k-1]}). \tag{9}$$

The average is obtained by weighting the distance of each pixel by its absolute RGB luminance y, as brighter pixels ought to contribute

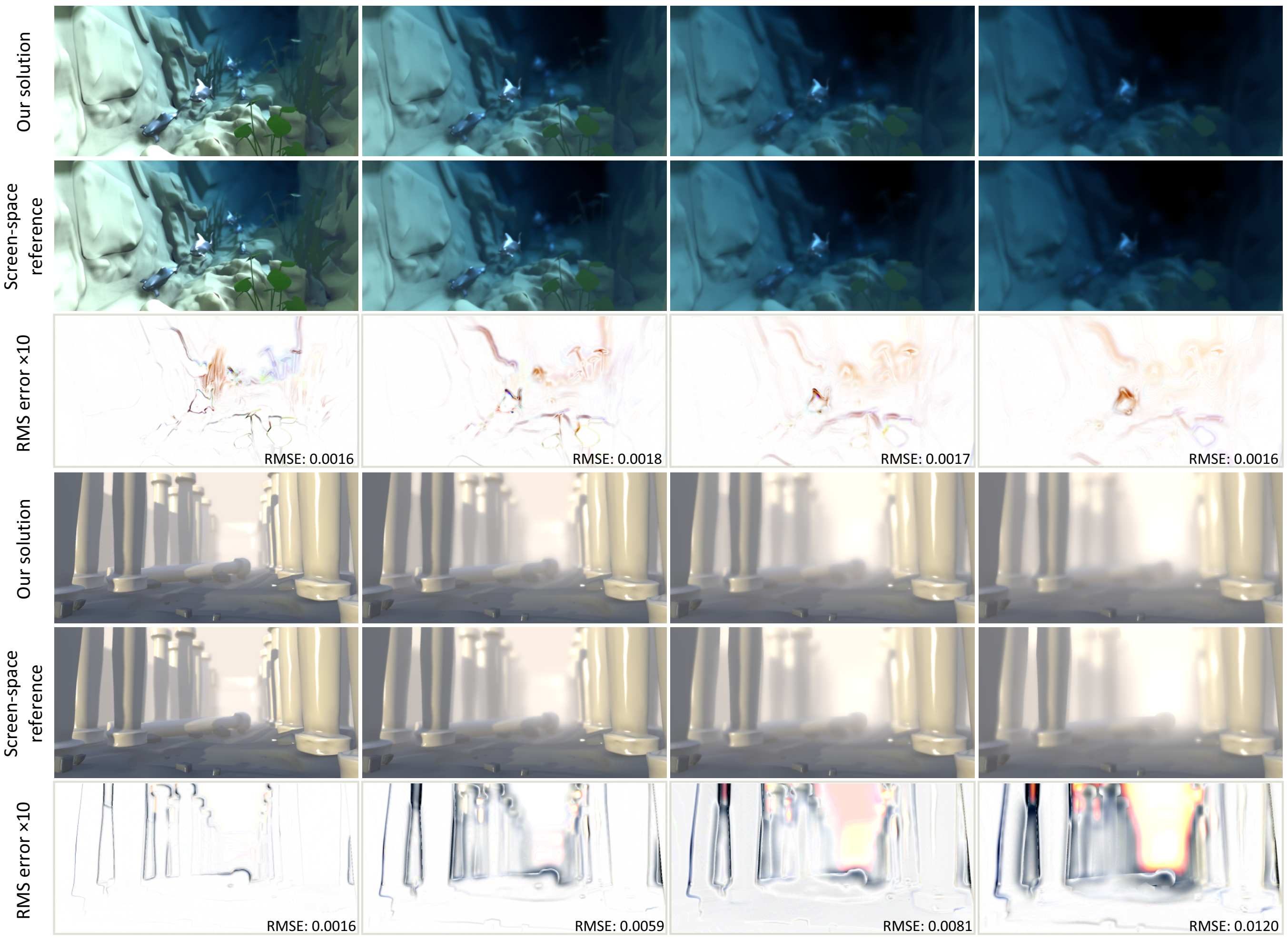

Figure 7: *Evaluation of our method for different values of medium optical thickness ($\tau \approx 2, 4, 6$ and 8 from left to right). Numerical evaluation against the screen-space reference is included to gauge the performance of our filtering solution under different optical conditions.*

more to the average distance that will be used as a masking criterion for obtaining the next MIP level (since *their* energy leakage we need to avoid primarily). In other words, the brighter an area is the more precise will our approximation at that position be.

We mathematically describe the averaging process in Eq. 8 as weighting W by Y and then performing the averaging as a convolution with the box filter U. The uniform distribution is used instead of a Gaussian in this case, because for the average spread-space distance there is no reason to favour the central samples – we only need to take into account the pixels which can potentially cause the leaking.

The effectiveness of our formulation in reducing the light leaking is shown in Fig. 6b–d and the overall comparison to the screen-space reference is provided in Fig. 7.

3.4 Final compositing

Having the filtered MIP map that represents the scattering computed, obtaining J essentially amounts to reading each pixel's corresponding distance-dependent MIP level from $\mathsf{I}^{[0..K]}$ and adding it to I'.

The mapping from a pixel's distance to its MIP level needs to take into account that both the corresponding filter size and the masking threshold T grow exponentially with the increasing MIP level (Sec. 3.3). Therefore, the corresponding MIP level will grow logarithmically with spread-space distance as

$$\ell(\mathsf{D}) = \operatorname{clamp}\left(\log_2 \frac{W(\mathsf{D})}{c}, 0, K\right) \qquad (10)$$

The scaling constant c determines the actual width of the corresponding filter in the MIP chain. Burt [1981] uses $c = 0.56$ for a 5×5 incremental filter, we however found that the value of 0.8 matches the screen-space reference solution much better in the case of our 4×4 filter. And, since the MIP levels must closely correspond to the masking threshold applied to them, c is also included in the definition of T in Sec. 3.3.

The issue with the direct application of Eq. 10 to obtain the appropriate MIP level is the emergence of discontinuities in the resulting image, as shown in Fig. 6a–d. This problem is inherent to pyramidal convolution methods. The common workaround to this problem first introduced in [Bertalmio et al. 2004] is to create a uniformly blurred version of the distance map and use this to determine the appropriate MIP level instead. This creates smooth transitions around depth discontinuities and therefore also gradual increase of the blurring level in these places.

The simplest way to perform this is applying an isotropic Gaussian pyramidal blur to D. This is very fast as the isotropic Gaussian filter

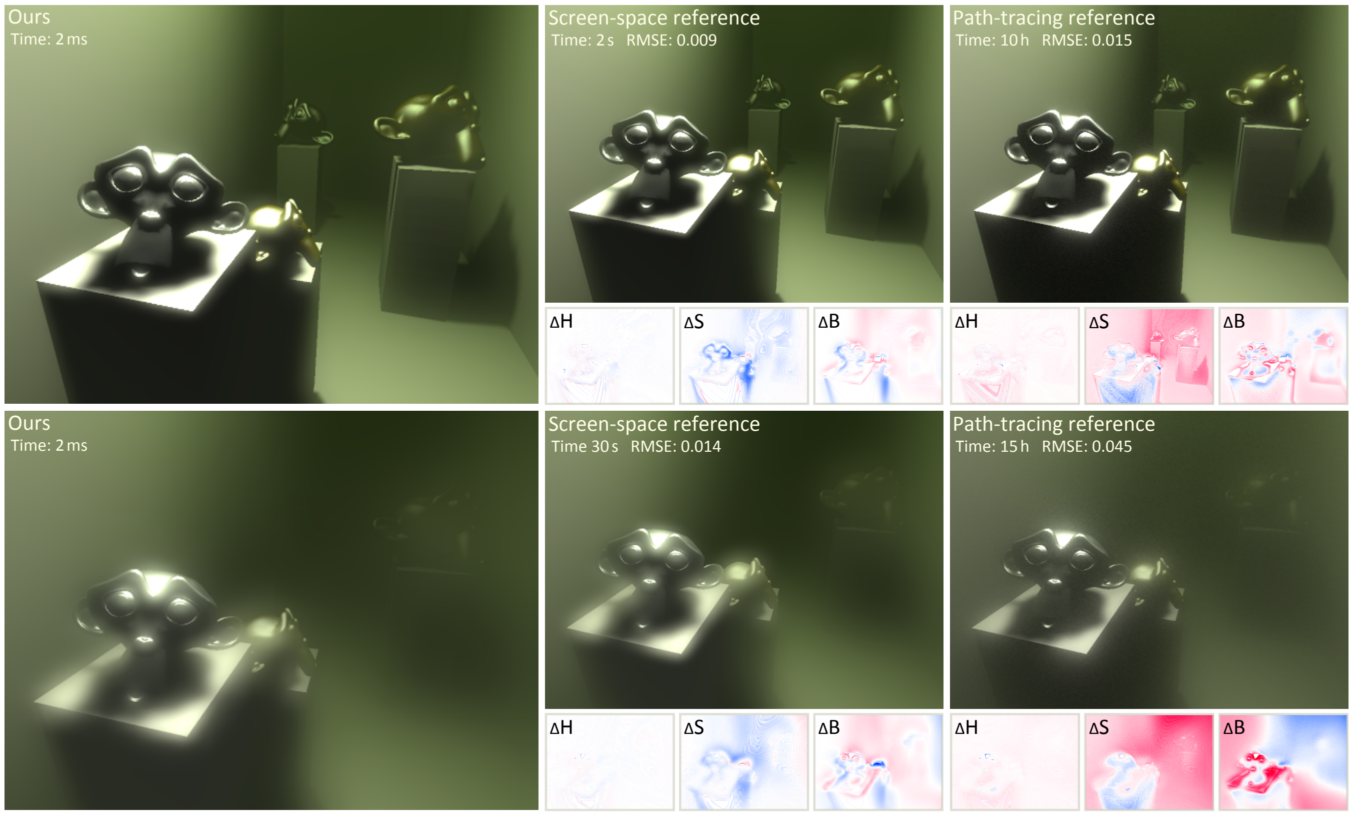

Figure 8: *Comparison to screen-space and path-tracing solutions for two different scattering intensities. We measured the average RMS error for RGB intensities and also per-pixel differences in HSB space. The peak error the HSB difference images represent is 20 %.*

is separable and D is a single-channel image. However the main flaw of simply using an uniformly blurred distance map is that the areas closer to the camera are blurred just as much as distant areas. Our proposal is based on the intuitive observation that the size of the blurring transition at distance discontinuities should be proportional to the blurring due to the scattering itself. This translates to using the distance of a pixel to determine the strength of blurring the distance map, and use *this* blurred distance value to calculate the query to the scattering MIP map and obtain the final image, J:

$$\mathsf{L} = \ell(\mathsf{D}^{[\ell(\mathsf{D}^{[0]})]}) \quad (11)$$

$$\mathsf{J} = \mathsf{I}' + \mathsf{I}^{[\mathsf{L}]}. \quad (12)$$

Our solution is similar to [Lee et al. 2009], but simpler. This is possible because the blurring caused by light scattering increases monotonically with the distance from camera, while in depth-of-field simulation it increases in both directions with increasing distance from the focal plane. Therefore, instead of building the specialized anisotropic MIP map described by Lee et al. we build a simple isotropic MIP pyramid, which can further be utilized by other screen-space methods, may the need be.

3.5 Implementation

In this section we provide additional performance considerations about the algorithm described above. First, it should be noted that it is not always necessary to build the full MIP chain of I (i. e., down to the 1-pixel level). If the maximal distance in the scene is known, then the value of the function W can be bounded with this distance and the used medium parameters. With the bound of W we can then determine the highest necessary MIP level K in the same way as ℓ is obtained in Eq. 10. In our implementation $K = 6$ proved to be sufficient in all cases.

Additionally, as an optimization we pack the spread space distances W into the alpha channel of the radiance texture when constructing the MIP map of I. Only one MIP map is therefore constructed, just its RGB channels are computed differently than its alpha channel.

Finally, to read $\mathsf{I}^{[0..K]}$ in Eq. 12 we use a linear-bi-cubic fetching operation, since we need to obtain an as smooth result as possible. This means that to obtain a value at level ℓ (which is generally not an integer as defined by Eq. 10) we read the two adjacent levels $\lfloor \ell \rfloor$ and $\lfloor \ell \rfloor + 1$, each with a bi-cubic fetch, and then linearly combine the results. This would require reading 32 samples in total, leading to an excessive cost of this step. We therefore modify the bi-cubic fetch in the way that groups of 4 pixels are fetched at the same time by a single bi-linear fetch from a position obtained by relating the respective pixels' weights given by the bi-cubic filtering function [Sigg and Hadwiger 2005]. We therefore decrease the number of fetches to 8 (linear ones), which on GPU are virtually as fast as point fetches. We also experimented with the circular filtering described in [Lee et al. 2009], but found it to perform much worse in our context, in spite of the same number of fetches it requires.

4 Results

We implemented the described algorithm in an interactive rendering framework written in C++ and GLSL, using OpenGL. The GPU used in all tests and examples was NVidia GeForce GTX 485 Mobile with 2 GB video memory.

Figs. 1, 7 and 8 show typical results of our approach. Fig. 9 shows

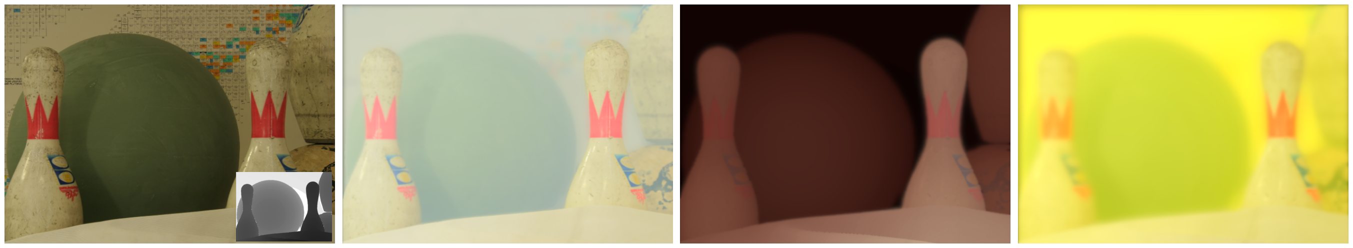

Figure 9: *Application of our technique to a photograph with an automatically segmented distance map* (inset) *using three different media.*

the application of our approach to an RGBZ photograph. Such images are likely to become increasingly popular in the future, with depth sensors such as Microsoft Kinect becoming commonly available to the mass market.

A quantitative evaluation of our approach is shown in Fig. 8. In addition to the screen-space reference we also compare to a path-tracing solution. Please note that we applied path tracing just to the scattered portion of the illumination, instead of the full global illumination. The reason for this is to eliminate differences caused by the traditional approximations to global illumination on surfaces that we utilize, such as soft shadow maps or ambient occlusion. It can be seen that despite numerous approximations our algorithm builds on (cf. Sec. 5), we are still able to obtain reasonable results even for very dense media.

The performance of our technique is detailed in Table 1; as can be seen it is always in hundreds of FPS, making it a good candidate for a post-processing step integrated in a real-time graphics engine. Probably the most important observation is that the computation time scales linearly with the number of rendered pixels, as opposed to the quadratic growth in the screen-space reference solution. In addition, the measurements show that using our anisotropic filter incurs only a very small additional cost in comparison to the naïve solution, despite increasing the quality considerably. Finally, we measured performance for different filtering kernel sizes; please refer to Sec. 5 for a discussion in this regard.

I/D Res.	Step 1	Step 2				Step 3	Total
		Ours			Naïve		
		2×2	4×4	6×6	4×4		
1 MPx	0.3 ms	0.7 ms	1.1 ms	2.6 ms	1.0 ms	1.2 ms	2.6 ms
2 MPx	0.6 ms	1.2 ms	1.6 ms	4.6 ms	1.5 ms	2.1 ms	4.3 ms
4 MPx	1.2 ms	1.9 ms	2.6 ms	8.6 ms	2.3 ms	4.2 ms	8.0 ms

Table 1: *Performance our our technique for different image resolutions. Each step is measured individually, Step 2 for different filter types. The total time is the sum of Steps 1–3 with our 4×4 kernel used in Step 2.*

5 Discussion

The approach described in this paper is an addition to the increasing number of screen-space methods, the history, strengths and weaknesses of which are briefly summarized in Sidebar 2.

Relation to depth-of-field rendering As already mentioned our method closely relates to [Lee et al. 2009], and some of the involved steps might be regarded as adaptations from this method. Further, similar to depth-of-field methods, an explicit bilateral filtering could be used to limit the illumination leakage even more (although our anisotropic filter in fact acts as a simple bilateral filter). Finally, an interesting possibility to explore is to use the scattering MIP map to render depth-of-field effects as well. This should be well feasible, and the main step towards such approach would be to combine the degrees of blurring due to scattering and de-focus and use the result to query the filtering hierarchy.

Filtering kernel size As the resolution of the MIP map levels halves with each additional level, the size of the filtering kernel must be even. Taking into account the results of [Burt 1981], we found that the filter size which provides the best compromise between price and quality is 4×4. While it is possible to use a 2×2 filter (with an approximately 30% performance gain in the filtering step, cf. Table 1), its application does not correspond to a Gaussian filtering anymore, as it is equivalent to a box filter. Its limited spatial support also causes disturbing temporal flickering artefacts, especially in scenes with a very high dynamic range. On the other hand, using a 6×6 kernel increases the cost of the filtering step considerably, roughly in proportion to the increase of fetches required to compute it. Though slightly reducing the temporal flickering, we found that it does not compensate for the increased computational costs.

5.1 Limitations

Approximation by PSF Approximating scattering or any other global effect in this way is inherently limited. First, the derivation of the underlying PSF itself (Eq. 2) relies on certain simplifying assumptions [Premože et al. 2004]. Although the derivation considers multiple scattering, the PSF expression is limited to homogeneous media. It is still possible to include heterogeneous media in an approximate way, by using ray-marching to integrate the medium parameters for each pixel individually; this would however increase the cost of our approach significantly.

Using the image of the scene as a starting point for approximating the scattering is also not entirely correct. The reason is that the calculation of the illumination even before considering the camera should already take the presence of a participating medium into account, which typically is not the case. Thus, incorporating at least some of the effects of scattering into the standard interactive rendering pipelines and algorithms is indeed an interesting direction for future exploration.

Another limitation is that light sources which are not directly visible are not accounted for. It would be possible to use more advanced techniques to remedy this, such as depth peeling to resolve occluded light sources and analytical airlight models to account for off-screen sources of light, but again, with considerable increases of the rendering time. The simple measure we use in the majority of our examples is the addition of an emissive term derived from the intensity of the environmental illumination in the scene. Also, as the employed model simulates multiple scattering as a blurring operation, high-frequency effects due to both on- and off-screen sources are not captured by our technique. These include phenomena such as light shafts and volumetric caustics, which in scenes

Sidebar 2: Screen Space

Approximating costly computation of 3D illumination effects by simple 2D image operations becomes increasingly popular in practical interactive applications such as computer games. The key idea behind such *screen-space* methods is that modern GPUs produce more than just RGB colour when making use of the deferred shading technique [Deering et al. 1988]. Often per-pixel depth, normal or reflectance information complements the colour framebuffer (Fig. 1). Consequently, screen-space techniques utilize this additional information and, parallel over all pixels, compute advanced shading effects from it.

Ambient occlusion [Mittring 2007] was one of the first effects to be simulated in screen-space, followed by later extensions to directional occlusion and indirect light [Ritschel et al. 2009] or sub-surface scattering [Jimenez et al. 2009]. Besides light transport, approximations to depth-of-field [Rokita 1993; Lee et al. 2009], glare [Kawase 2005] or motion blur [Ritchie et al. 2010] are of practical importance.

All screen-space methods face the difficulty of balancing between fast and simple but approximate local linear filtering and complex non-linear filtering that has to account for a high number of neighbouring pixels. Furthermore, PSFs define how much a pixel contributes to other pixels (called a „scatter-type" computation). This is not identical to a function that encodes for every pixel how much to „gather" from its neighbours. Nonetheless, scattering-type computation is routinely replaced by gathering ones as they fit the execution on modern GPUs much better.

However, large linear filters still remain costly. Every pixel has to iterate over a large neighbourhood, and, in the extreme case of large PSFs each pixel enumerates all pixels of the entire screen. A substantial acceleration is possible by using (Gaussian) MIP maps. After a MIP map of the framebuffer is constructed arbitrary-sized Gaussian filters can be applied to it in constant time. While MIP maps work well, e. g., for glare [Kawase 2005], they have difficulties if the gathering to perform is significantly different from a Gaussian. Plus, even if the PSF is a Gaussian, the gathering function is not necessarily a Gaussian anymore. One solution of this problem are anisotropic MIP maps suggested by Lee et al. [2009]. Such MIP maps contain only values that actually would contribute to the result, depending on an additional criterion such as depth. Alternatively, the computation can be accelerated using stochastic sub-sampling [Jimenez et al. 2009] or separable filtering [Huang et al. 2011]. Diagonal structures or high-dynamic-range content are however likely to cause artefacts with both simplifications.

The biggest limitation shared by all scree-space methods is that objects or lights which are not visible in the framebuffer do not contribute to the image. However, in the years to come, screen-space methods—due to their efficiency, ease of implementation and control—are likely to remain the solution of choice until the full simulation of distributed effects [Cook et al. 1984] becomes feasible.

Screen Space References

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. *SIGGRAPH Comput. Graph. 18*, 3, 137–145.

DEERING, M., WINNER, S., SCHEDIWY, B., DUFFY, C., AND HUNT, N. 1988. The triangle processor and normal vector shader: a VLSI system for high performance graphics. *SIGGRAPH Comput. Graph. 22*, 4, 21–30.

HUANG, J., BOUBEKEUR, T., RITSCHEL, T., HOLLNDER, M., AND EISEMANN, E. 2011. Separable approximation of ambient occlusion. In *Eurographics 2011 - Short papers*.

JIMENEZ, J., SUNDSTEDT, V., AND GUTIERREZ, D. 2009. Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception 6*, 4.

KAWASE, M. 2005. Practical Implementation of High Dynamic Range Rendering. In *Game Developers Conference*.

MITTRING, M. 2007. Finding next gen: CryEngine 2. In *SIGGRAPH courses*.

RITCHIE, M., MODERN, G., AND MITCHELL, K. 2010. Split second motion blur. In *SIGGRAPH Talks*.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in screen space. In *Proc. I3D*.

with high dynamic range are often important to convey the presence of a participating medium. Given the character of our method, it is clear that simulating these requires employing an additional, complementary approach, such as the fast epipolar sampling technique by [Engelhardt and Dachsbacher 2010].

The method also requires specification of one arbitrary parameter, aside from the physical properties of the simulated medium. This is the threshold width ε (Sec. 3.3). However, it is an intuitive quantity and once configured for a given scene it can in most cases remain unchanged.

Decoupling of scattering and absorption Despite the fact that absorption and scattering are intertwined effects the former has to be taken care of before the latter in a screen-space method like ours. The reason is that absorption is a distance-dependent effect that can produce subtle but perceivable hue changes even for small distance variations. As such it needs to be calculated in the full-screen resolution where the distance information per-pixel is the most accurate, hence is can not be done in Step 2 where the lower-resolution MIP map is constructed. However, it also can not be calculated *after* the filtering phase, because the blurred radiance coming from different distances will exhibit different levels of absorption; if one would try to compute the absorption after first blurring the scattered radiance, multiple 'halos' from different distances could potentially overlap in any given pixel, but each pixel would be attenuated depending on *its own* distance, which might not correspond to distances the halos originate from.

Hierarchical convolution The approximation of an incremental convolution by a hierarchical one also bears some limitations, especially if an anisotropic filter is used as in our case. Although the lower-resolution MIP levels are sufficient to represent the blurred information from the perspective of its frequency, some spatial or temporal high-frequency artefacts can still appear, mostly due to the fact that the distance information in the higher MIP levels is averaged. This might become a problem in the presence of a high-frequency illumination, or in case of very dense media.

6 Conclusion

Our approach allows to approximate distinct colouring and blur effects that can be essential to convincingly depict participating media such as water or various atmospheric conditions. As described above, it shares the usual shortcomings of screen-space approaches, in that it does not account for light not present in the framebuffer, but also all their beneficial properties, like fast speed—taking only a few milliseconds for typical screen resolutions—and its ease of implementation and integration into an existing screen space / deferred shading-pipeline. We deem this trade-off as acceptable and hope for an application of our technique in games and other real-time applications, but also image editing tools.

Acknowledgements We thank Bernhard Reinert for proofreading and the anonymous reviewers for their feedback. This work has been supported by the Max Planck Center for Visual Computing and Communication.

References

BERTALMIO, M., FORT, P., AND SANCHEZ-CRESPO, D. 2004. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *Proc. 3DPVT*.

BURT, P. J. 1981. Fast filter transforms for image processing. *Computer Graphics and Image Processing 16*, 20–51.

ENGELHARDT, T., AND DACHSBACHER, C. 2010. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proc. SIGGRAPH*.

LEE, S., KIM, G. J., AND CHOI, S. 2009. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Transactions on Visualization and Computer Graphics 15*, 453–464.

LOPEZ-MORENO, J., CABANES, A., AND GUTIERREZ, D. 2008. Image-based participating media. In *Proc. CEIG*.

NARASIMHAN, S. G., AND NAYAR, S. K. 2003. Shedding light on the weather. In *Proc. CVPR*.

PREMOZE, S., ASHIKHMIN, M., AND SHIRLEY, P. 2003. Path integration for light transport in volumes. In *Proc. EGRW*.

PREMOŽE, S., ASHIKHMIN, M., RAMAMOORTHI, R., AND NAYAR, S. K. 2004. Practical rendering of multiple scattering effects in participating media. In *Rendering Techniques*.

ROKITA, P. 1993. Fast generation of depth of field effects in computer graphics. *Computers & Graphics 17*, 593–595.

SIGG, C., AND HADWIGER, M. 2005. *GPU Gems 2*. Addison-Wesley, ch. 20.

Oskar Elek received his MS degree in 2011 at the Charles University, Prague and is currently pursuing his PhD at the MPI Informatik, Saarbrücken. His main research interests are efficient physically-based rendering on GPUs and simulation of light scattering in participating media. Contact him at `oelek@mpii.de`.

Tobias Ritschel is a junior research group leader in the Max Planck Center for Visual Computing and Communication at the MPI Informatik. His interests include interactive and non-photorealistic rendering, human perception and data-driven graphics. He received the Eurographics PhD dissertation award in 2011. Contact him at `ritschel@mpii.de`.

Hans-Peter Seidel is the scientific director and chair of the Computer Graphics Group at the MPI Informatik and a professor of computer science at Saarland University. In 2003, he received the Leibniz Preis, the most prestigious German research award, from the German Research Foundation (DFG). He is the first computer graphics researcher to receive such an award. Contact him at `hpseidel@mpi-sb.mpg.de`.