Expressive Single Scattering for Light Shaft Stylization

Kol, Timothy R.; Klehm, Oliver; Seidel, Hans-Peter; Eisemann, Elmar

DOI
10.1109/TVCG.2016.2554114

Publication date
2017

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Visualization and Computer Graphics

Expressive Single Scattering for Light Shaft Stylization

Timothy R. Kol, Oliver Klehm, Hans-Peter Seidel, Elmar Eisemann

Fig. 1. Example of our stylized scattering. Left: physically correct single scattering using the original occluders. The leaves of the tree block most of the light, causing a rather subtle effect. Right: stylized scattering with *occluder manipulation*. Using our system, an artist can easily add holes into the shadow map of the tree, producing more pronounced scattering effects. While physically incorrect, it is not obvious for the viewer that the right image uses fake occlusion information. Surface shadows are created from the original shadow map.

Abstract—Light scattering in participating media is a natural phenomenon that is increasingly featured in movies and games, as it is visually pleasing and lends realism to a scene. In art, it may further be used to express a certain mood or emphasize objects. Here, artists often rely on stylization when creating scattering effects, not only because of the complexity of physically correct scattering, but also to increase expressiveness. Little research, however, focuses on artistically influencing the simulation of the scattering process in a virtual 3D scene. We propose novel stylization techniques, enabling artists to change the appearance of single scattering effects such as light shafts. Users can add, remove, or enhance light shafts using occluder manipulation. The colors of the light shafts can be stylized and animated using easily modifiable transfer functions. Alternatively, our system can optimize a light map given a simple user input for a number of desired views in the 3D world. Finally, we enable artists to control the heterogeneity of the underlying medium. Our stylized scattering solution is easy to use and compatible with standard rendering pipelines. It works for animated scenes and can be executed in real time to provide the artist with quick feedback.

Index Terms—Interactive stylization, artist control, single scattering.

1 Introduction

The *scattering* of light is a natural phenomenon that can drastically change the look of a scene. It occurs when light travels through a *participating medium*, such as air, where it can interact with particles. Among the most dominant effects caused by scattering are crepuscular rays, which are best described as being no more than *light shafts*. Light shafts not only add realism and spatial cues for a better scene understanding, but can also serve artistic purposes. They are visually pleasing, and therefore frequently found in art pieces, where they have a history of being *stylized* rather than being physically correct.

- *T. R. Kol and E. Eisemann are with the Computer Graphics & Visualization group at the Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands.*
  E-mail: {t.r.kol, e.eisemann}@tudelft.nl.
- *O. Klehm and H.-P. Seidel are with the Department of Computer Graphics, MPI Informatik, Saarbrücken, Germany.*
  E-mail: {oklehm, hpseidel}@mpi-inf.mpg.de.

Figure 2 shows two pieces of art that illustrate some ways in which scattering can be stylized. In the left image, light shafts are used to emphasize the sailboat, a technique that is common in comics and animation movies to highlight a focus object. For this painting, the harshness of the lighting is particularly striking, as most real-world light shaft boundaries are rather smooth. In this sense there is a clear contrast between the upper light shaft, that exhibits a smooth behavior, and the lower part, which is more pronounced and clearly highlights the object of interest.

In the right image, light shafts seem to be added and removed at will, taking little note of solid objects that would normally block the light, such as the group of trees on the left. The color of the light shafts seems to vary between green and yellow hues in a physically incorrect but appealing manner. Another point of interest is that,

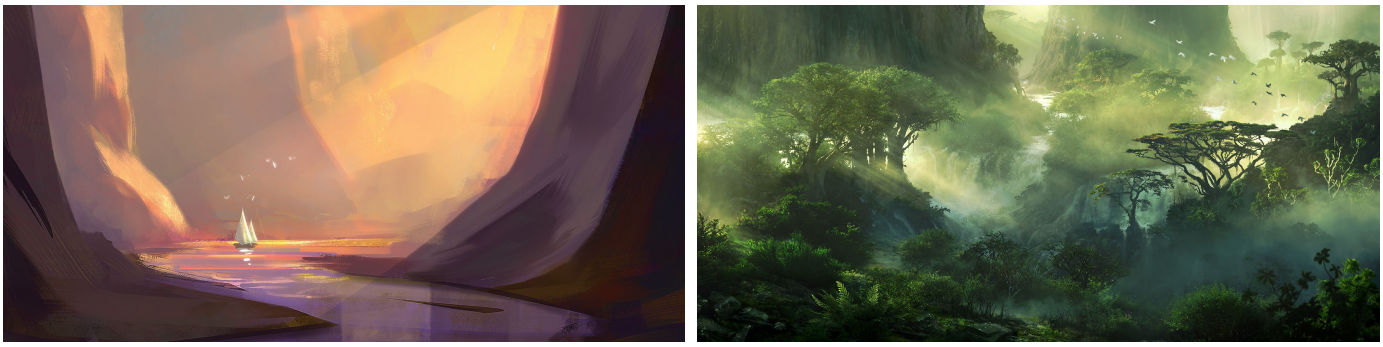

Fig. 2. Example of stylized scattering in (concept) art. Left: the light shafts have a very sharp boundary in order to emphasize the sailboat. Source: Roberto Gatto (www.robertogattoart.com); used with permission. Right: light shafts seem to start and stop in mid-air for no clear reason other than aesthetics. Source: Jonas De Ro (www.jonasdero.be); used with permission.

unlike in the image on the left, the light shafts have a rather irregular appearance. While perhaps somewhat exaggerated here, this effect can also occur in nature due to a medium being *heterogeneous*, which means that the scattering particles are not uniformly distributed, creating a spatially varying density. Note that both images look plausible and visually pleasing, despite the physically incorrect light shafts and the fact that these do not correlate with surface shadows. It can be concluded that for artists, stylization of light shafts is more of a rule than an exception, and serves an important role in many artworks. Further, stylized scattering does not necessarily produce confusing or implausible results. Such manipulations seem widely accepted, and may not even be noticed by the untrained observer.

Since the dawn of computer graphics, however, most research in scattering has focused on efficiently obtaining *physically correct* and *realistic* images. For a realistic scattering process, light shafts manifest as follows; when light illuminates an optically thin participating medium like air, it often bounces off particles before reaching the eye. If a single bounce occurs, the process is called *single scattering*. Visually, the medium through which the light travels is lit. However, when part of the light is blocked by an object – which we call an *occluder* – before it bounces, it cannot be scattered towards the viewer and these areas appear darker.

Simulating single scattering in homogeneous media can be achieved in real time [1], [2]. As in nature, light shafts depend on the scene layout and not on any specific scene element, which makes stylization difficult as the appearance cannot be modified directly. In consequence, there is a need for specialized rendering techniques to enable direct and easy artistic control, but very few algorithms exists [3], [4].

Our work offers new ways of modifying light scattering to produce effects similar to the aforementioned stylization used in art. Hereby, we hope to give artists more freedom, enabling them to carry on the trend of stylized scattering from more traditional art to video games and film effects. We employ three manipulation concepts in our work. First, we introduce *occluder manipulation*, which allows the user to add and remove light shafts (Figure 2, right) or sharpen them (Figure 2, left) . Figure 1 illustrates the addition of light shafts, which leads to a brightened scene and additional details, influencing the appearance significantly. Second, our algorithm provides simple controls to achieve various styles, expressive color changes, and mood alterations in a scene. For this, we introduce two techniques: *transfer functions* and *light map optimization*. Third, we introduce a solution to control the heterogeneity of the medium by enabling interactive changes of the volume's properties.

As with most artistic tools, it is crucial that users receive interactive feedback to explore possible parameter choices. For this reason, we focus on real-time methods. Our high-level definitions enable the transfer of a general style to scenes with different geometry, camera, and light settings, and support animation. Hereby, we introduce effective means for controlling the scattering in 3D scenes for interactive applications.

Specifically, our work makes the following contributions:

- light shaft addition, removal, and enhancement using image-based occluder manipulation;
- light shaft color changes via user-editable transfer functions based on view ray properties, and light map optimization based on user-drawn strokes;
- light shaft irregularity by controlling the medium's heterogeneity with a 3D painting tool; and
- light shaft animation through dynamic occluder manipulation, and key-framed transfer functions for animated scenes.

This paper is an extension of our previous work on stylized scattering [5], but includes several novel elements. We make a better link to existing 2D art, strengthening our original motivation, and demonstrating the usefulness of our techniques via various new results. We also introduce the control of heterogeneity, which is a novel concept, as we previously considered homogeneous media only. Furthermore, we present a new light map optimization, which is an alternative to transfer functions and enables varying light shaft colors, while maintaining physical plausibility.

2 Related Work

Cinematic relighting [6], [7] generally gives artists the possibility to predict the final rendering in order to support them in tasks such as lighting design and material definitions. However, the underlying calculations in these systems are usually physically-based, and the possibilities for abstraction and stylization are often restricted to the scene setup.

2.1 General Stylization

In recent years, solutions to influence physics for the purpose of expressiveness have received increasing attention and there are several approaches to stylize natural phenomena. Modifications of the light transport [8], [9], [10], shadows [11], [12], [13], [14], caustics [15], motion blur [16], or depth of field [17], have been proposed to significantly influence the appearance of a scene and to guide the observer to specific regions of interest. Similarly, other stylization techniques have been demonstrated for focus control [18]. For a more detailed exposition of similar work on appearance and lighting editing, we refer to a recent overview [19].

2.2 Stylized Scattering

Regarding scattering stylization, only a few approaches have been suggested. Artistic beams [3] let the user modify individual light rays by influencing shape, falloff, and color. The modifications are used to find a plausible, optimized mapping to properties of the scene's participating medium. Treating light rays individually can be an advantage, but global control becomes more time-consuming and difficult. In contrast, our stylization method uses parameters derived from scattering to directly map to the final scattering result.

One can also rely on a set of painted input images to find the optimal volume parameters to best match the provided target images [4]. In this case, the volume parameters are stored in a voxel grid, which limits the possible resolution and performance, despite the employment of an efficient process. Furthermore, defining the input requires a certain artistic skill, and the final rendering is bound by the actual physical process, limiting the potential expressiveness.

Furthermore, Hašan and Ramamoorthi [20] presented a method for efficiently re-rendering a scene for which the volume's single scattering albedo values have been modified. While stylized scattering is an application, their approach focuses rather on increasing the performance of re-rendering, and not so much on design and stylization tools. Moreover, they require a computationally expensive pre-process due to taking the full light transport for dense volumes into account, while we aim for real-time solutions for single scattering in optically thin media.

In our solution, we want to make it easy to define plausible results, but also enable more expressive solutions, potentially leaving the physical behavior. For this, efficient computation is key to allow artists to rapidly explore various options. Hence, we focus on existing real-time solutions for single scattering. There are several options for efficient computation of single scattering; min-max mipmaps [1], voxelized shadow volumes [21], shadow volumes based on shadow maps [22], or prefiltered single scattering [2]. While approaches for multiple scattering exist [23], their precision is still relatively low due to the use of a coarse grid, which is why we concentrate on single scattering only.

2.3 Specific Techniques

Occluder manipulation as a possible means of stylization has been applied before in the form of proxy geometries to modify light transport in a scene. Schmidt et al. [24] introduced the idea of path-proxy linking, defining invisible copies of scene objects, which are modified using affine transformations and only affect a certain individual component of the light transport, such as shadows. While we also modify occluders to change scattering behavior, we propose specialized and parameterizable manipulation methods that are useful for stylization. To this end, we manipulate occluders using morphological operators and work in the space of a 2D shadow map as this efficiently gives direct control over creating, removing, and enhancing light shafts.

A change in color is often achieved by the use of transfer functions, such as for surface shading [25] or volume visualization [26]. Unlike in volume visualization [26], we do not input medium properties at a point in space, but evaluate scattering-related values along the view ray as parameters for our transfer functions. Alternatively, we propose a light map optimization that employs inverse rendering, a general concept that derives scene parameters from 2D user input. Schoeneman et al. [27] were first to optimize light properties (intensity and color) in a least-squares sense to satisfy user-defined target images. Klehm et al. [28] applied the concept to environmental illumination of heterogeneous media. We focus on the special case of thin homogeneous media and optimize a light map for a directional light source. In contrast to the aforementioned work, users only draw strokes to define light shaft colors instead of a full image.

Due to performance constraints, heterogeneous media have received limited attention for real-time applications. Zhou et al. [29] proposed a composition of simple radial basis functions. This approach, despite limiting the amount of detail of the heterogeneity, allows for easy designing of a heterogeneous medium by placing the radial basis functions with the aid of a brush and eraser. However, the focus lies on an approximation of the physically correct result and uses an analytic model to evaluate the basis functions. Instead of designing a volumetric element, we use heterogeneity to locally enhance the indirect effect of light shafts.

3 Real-time Scattering Background

Before discussing our algorithm, we will first give a brief introduction to single scattering.

Radiance caused by single scattering from a single directional light towards a camera at $\mathbf{x}$ from direction ω_i is computed by integrating the view ray up to the first visible surface at distance s:

$$L_\mathrm{scat}(\mathbf{x},\omega_\mathrm{i}) = \sigma_\mathrm{t}\,\rho\,f\int_0^s e^{-t\sigma_\mathrm{t}}\,V(\mathbf{x}_t)\widetilde{L_\mathrm{i}}(\mathbf{x}_t)\,dt, \qquad (1)$$

assuming a homogeneous medium with extinction coefficient σ_t, scattering albedo ρ, and constant phase function f. $\widetilde{L_\mathrm{i}}(\mathbf{x}_t)$ denotes the unoccluded, incoming light at the scattering point and $V(\mathbf{x}_t)$ the corresponding visibility from the light source. Note that we skip attenuation from the light source to the sampling point $\mathbf{x}_t$, as this results in complete attenuation for a directional light source located at infinity.

A common approximation is to factor out visibility [1], [2], [21], [30], which we also identify as a useful parameter for stylization purposes. The equation then becomes:

$$L_\mathrm{scat}(\mathbf{x},\omega_\mathrm{i}) = \sigma_\mathrm{t}\,\rho\,f\int_0^s e^{-t\sigma_\mathrm{t}}\widetilde{L_\mathrm{i}}(\mathbf{x}_t)\,dt\;\; s^{-1}\int_0^s V(\mathbf{x}_t)\,dt. \qquad (2)$$

The first integral can be computed analytically [31] for a variety of light sources and is typically assumed to be constant for a directional light source. The second integral represents an average visibility and can be computed efficiently with an image-based solution, using a shadow map rendered from the light source, and a depth map rendered from the camera. Given a pixel in the image from the camera, its underlying depth value (distance to surface s) and point $\mathbf{x}$ define a segment in space, along which the visibility should be integrated. Using a ray marching process on the shadow map along this segment, the light visibility $V(\mathbf{x}_t)$ for each of these positions can be tested with a simple shadow map lookup. While being conceptually simple, this approach is not very efficient. Various acceleration methods [1], [2], [21], [30] have been proposed, and we use the solutions by Chen et al. [1] and Klehm et al. [2], which work comparably well and are executed on a shadow map. In all examples, we use the Henyey-Greenstein phase function for scattering to increase the initial physical correctness. However, other choices (even a simple constant) would be valid, too.

4 Stylized Single Scattering

Our method consists of three major techniques to perform scattering stylization, which can also be combined. The first modifies occluders in order to influence the scattering appearance by enhancing, adding, or removing light shafts. The second technique consists in the colorization of light shafts. We enable the definition of a transfer function that can be used to drastically influence color, brightness, and contrast. We rely on values (e.g., scene depth) derived along the view ray to define the final output color. As an alternative to the transfer function, the light shaft colors can also be modified by a light map optimization, driven by strokes drawn by the user. The third technique focuses on giving the light shafts a more irregular look by using an efficient algorithm to approximate a heterogeneous medium. The heterogeneity can be manipulated in multiple ways using a 3D painting tool. In the following, we will give an overview of these three techniques.

4.1 Occluder Manipulation

The main observation is that the appearance of single scattering in a scene largely depends on the number, size, and contrast of light shafts. They become visible due to differences in the light visibility along neighboring view rays (i.e., screen pixels). These differences are often caused by openings in the occluder (i.e., holes through which the light can shine), such as a gap in the clouds. Our system enables artists to modify light shafts by editing the shadow map, which is used to capture the occluders in the scene. Here, we present the various modification options.

Hole Filling

Physically-based scattering can sometimes produce unwanted effects. For instance, while the left image in Figure 13 is visually pleasing, the tiny light shafts along the ground give a somewhat chaotic nature to this otherwise serene scene. To appreciate the image more, we would like to *remove* these distracting details, caused by small holes in the occluders. By closing holes, we reduce the emphasis of the palm trees and simplify the light shaft appearance.

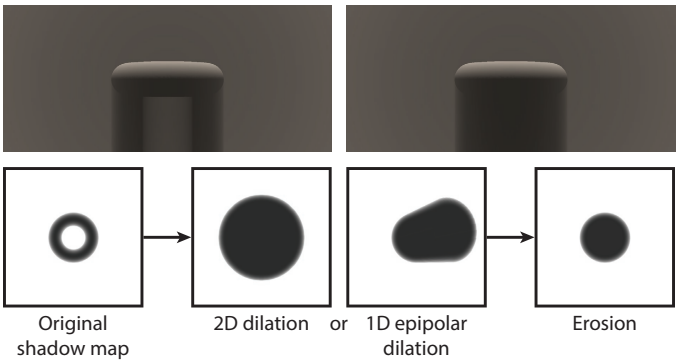

Fig. 3. Morphological filtering. Top: rendering a torus with directional light coming straight from the top with the unmodified (left) and hole-filled (right) shadow map. Bottom: shadow maps. From left to right: original shadow map; 2D dilation; 1D epipolar dilation; subsequent erosion, which for this particular scene yields identical results for both dilations.

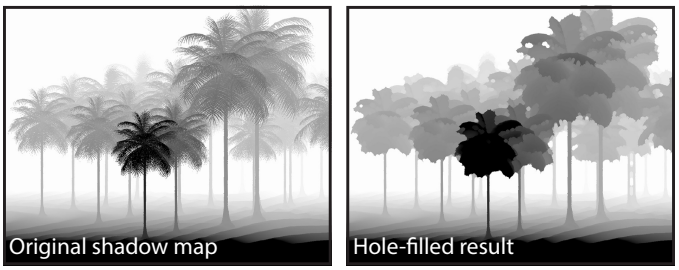

Fig. 4. Hole filling applied to the shadow map of the palm tree scene. Left: original shadow map. Right: result using a kernel size of 10.

For this hole filling, we make use of an image-based approach in which we directly modify the shadow map used for the scattering evaluation. One solution for hole filling is the use of 2D morphological filters. More precisely, a *closure* operation is applied, consisting of a *dilation* followed by an *erosion* (both are elementary operators). A dilation in the shadow map replaces a value by the minimum in a certain neighborhood, whereas an erosion replaces a value by the maximum. All holes that are removed in this way lead to the elimination of the corresponding light shafts. For illustration, an exaggerated example is shown in Figure 3, where we remove the entire hole in the torus.

One important factor is the neighborhood to be considered around each pixel, often referred to as the *filter kernel*. Traditionally, a square or circle is applied; per default, we use the latter. We let the user control the kernel size, which defines the strength of the hole filling process. The resulting shadow map for the palm tree scene is shown in Figure 4.

Silhouette Enhancement

We have shown how to remove details from the scattering result, but in some cases the opposite is desired. If scattering is very subtle, an artist may want more prominent light shafts; e.g., the middle image in Figure 13. To this extent, we can take an approach very similar to hole filling to *enhance* light shafts caused by the silhouette of an object. The idea is to extrude objects along the view rays, hereby increasing their thickness. We achieve the extrusion by a

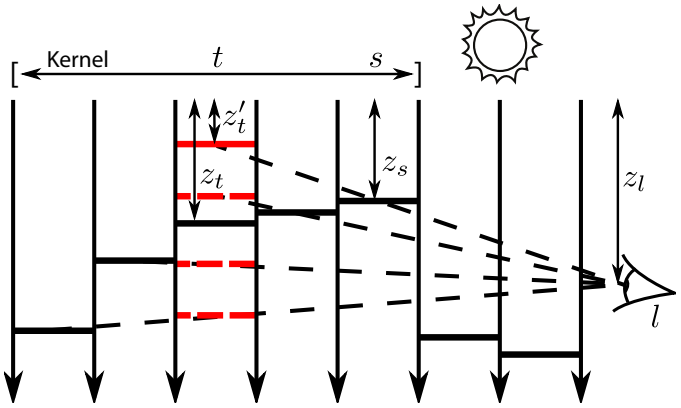

Fig. 5. 1D epipolar dilation. Areas between the parallel light rays represent texels, with the black horizontal bars denoting their depth in the shadow map. A 1D kernel (of size 5 in this example) is constructed for texel t with depth z_t. Given the epipole l with depth z_l, the whole kernel is sampled and Equation 3 is applied (red horizontal bars). The minimum value z'_t is in this case found for the sample s with depth z_s.

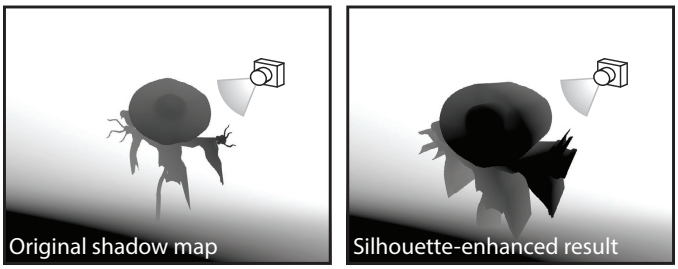

Fig. 6. Silhouette enhancement applied to the shadow map of the scarecrow scene. Left: original shadow map. Right: result using a kernel size of 100; the object is visibly extruded in the view direction.

1D dilation in the shadow map away from the epipole (the camera position projected onto the shadow map). Hereby, the occluder extension is always hidden by the object itself, as the camera only sees the first surface. However, the object's volumetric shadow is increased, as the extrusion is usually visible from the light source.

The *1D epipolar dilation* works as follows. For each texel t, we construct a 1D kernel along the line in the shadow map from t towards the epipole l. A standard dilation computes the minimum of all samples within the kernel; however, this does not satisfy the required extrusion from the camera position, as it effectively extrudes points in the plane orthogonal to the light direction. To solve this, we need to consider the changing z-coordinate along the view ray, which forms a sloped filter kernel. Thus, we modify the new depth z'_t of t given an input sample s as follows:

$$z'_t = \min(z_t, \frac{\mathrm{dis}_{2D}(t,l)}{\mathrm{dis}_{2D}(s,l)}(z_s - z_l) + z_l), \tag{3}$$

with $\mathrm{dis}_{2D}(t,l)$ denoting the 2D distance between texel t and epipole l as projected on the shadow map, with z_t and z_s the depth values of t and s in the shadow map, and z_l the z-coordinate of the epipole l. This results in a mix of z_s and z_l modulated by the ratio between the distances t to l and s to l. This value, as per the definition of a dilation, is used only if it is lower than the current lowest depth value z_t (hence, the $\min$). The process is illustrated in Figure 5.

When applied, the enhancement of the light shafts is twofold. First, the 1D epipolar dilation fills holes, removing small light shafts. Second, as the extrusion process is aligned with the view rays, it increases contrast between neighboring pixels, leading to sharper boundaries, as seen on the left in Figure 2. In consequence, occluders will block more light as they are effectively larger and the contrast difference between light shafts is pronounced. Figure 6 shows the resulting shadow map for the scarecrow scene.

Hole Creation

Silhouette enhancement makes the light shafts more prominent, but to obtain a visible effect, sufficient light shafts need to be present. Situations can occur where this is not the case, like in the right image in Figure 13. Here, only a few light shafts fall through the flowerbed, creating subtle scattering, no matter how much silhouette enhancement is applied. Yet an artist may want more light to burst through the flowers. For this reason, besides removal, we also offer a solution to *add* additional light shafts. In contrast to the previous hole filling operation, we instead create random holes, the result of which can also be seen in the teaser. Two simple steps are applied: we first generate a hole map and subsequently use it to modify the rendering of a shadow map.

The hole creation process is steered by various parameters to influence the average size, number, and density of the holes. In order to avoid perfectly uniform holes, for which the resulting light shafts can look too regular, we make use of Perlin noise [32]. By using a thresholding operation, we can transform it into a binary mask exhibiting randomization in shape, which we use as our hole map. Given some 2D texture coordinates t, the binary mask has a value of $H(t)$ given by

$$H(t) = \begin{cases} 0 & \text{if } N(tgf) \geq h \\ 1 & \text{otherwise} \end{cases}, \tag{4}$$

with h the user-defined hole probability or threshold value, f the user-defined frequency, g the Perlin noise gradient grid size, and $N(q)$, for $q = tgf$, given by

$$N(q) = \sum_{i=0}^{o-1} \frac{p^i P(q)}{\max(1, o-i-1)}, \tag{5}$$

where o is the number of user-defined octaves, p the user-defined persistence and $P(q)$ the classical 2D Perlin noise. The process is illustrated in Figure 7, where Perlin noise is created with parameters $f = 0.3$, $p = 0.5$, and $o = 5$, which is then thresholded using a hole probability $h = 0.5$, which results in the displayed hole map.

There are several ways to apply the hole map to influence visibility queries. Typically, we want to limit the effect of holes to pre-defined objects and the following solutions are only applied to those objects. One way is to discard fragments in the shader when rendering the shadow map if they belong to one of the pre-defined objects and the hole map value is 1 at their location. Alternatively, we render the objects in a separate shadow map and apply a $\max$ composition with the hole map. Basically, a hole is defined by pushing the depth values at a certain location to 1, i.e., the far plane. However, in this case shading calculations have

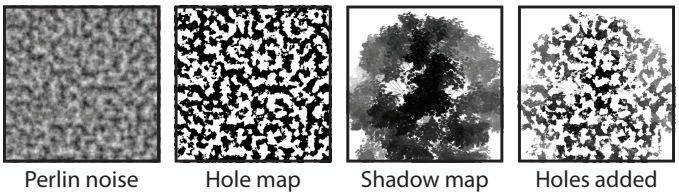

Fig. 7. Hole creation results are shown in Figure 1. Left to right: Perlin noise with a user-defined frequency; thresholding with a user-defined hole probability; original shadow map of the tree; shadow map with holes created from the hole map.

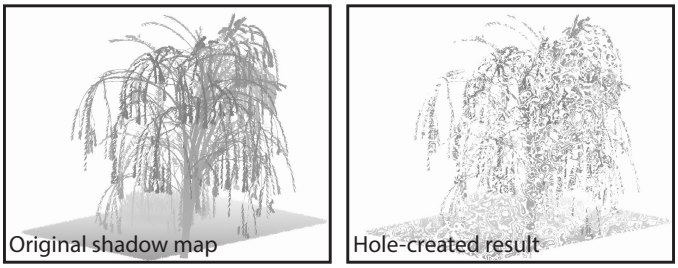

Fig. 8. Animated hole creation applied to the shadow map of the flowerbed scene. Left: original shadow map. Right: resulting shadow map. Note that in this case the hole creation was also applied on the ground, which may give strange results depending on the viewpoint. As mentioned before, we simply solve this by using two shadow maps and compositing them later on when necessary.

to test visibility against two shadow maps. While the first method is more elegant, the latter may be easier to integrate into an existing pipeline and makes a local adaptation of resolution per object possible.

Using Perlin noise naturally enables animation. We use a two-step lookup; first, a time-based offset is used to rotate the sample point before the lookup. We then use the resulting noise value as an offset to the original sample point and perform an additional noise lookup. Effectively, we apply a time-dependent noise to the lookup position, which causes an apparent global movement of the holes, but also a change in their size and structure. Hereby, a time-based rotation proved to give appealing results, while keeping the appearance consistent. Using this technique, we can simulate the effect of leaves moving in the wind or fake the notion of movement through participating media, which we show in our supplemental video. Figure 8 shows the result when applied to the shadow map of the flowerbed scene.

While this method might seem mostly unsuitable for solid objects, under certain conditions, e.g., to emphasize a character or simulate motion (Figure 9), it can still be useful. The main application area is still on less recognizable shapes, such as foliage or other detailed geometry.

4.2 Color Modifications

Typically, the resulting color of volumetric scattering is defined by the color of the light source or albedo of the scattering volume (see Equation 1). We propose two orthogonal techniques that enable users to colorize light shafts individually: transfer functions and optimized light maps. Both approaches target a goal-driven editing experience, where the user directly specifies color sets.

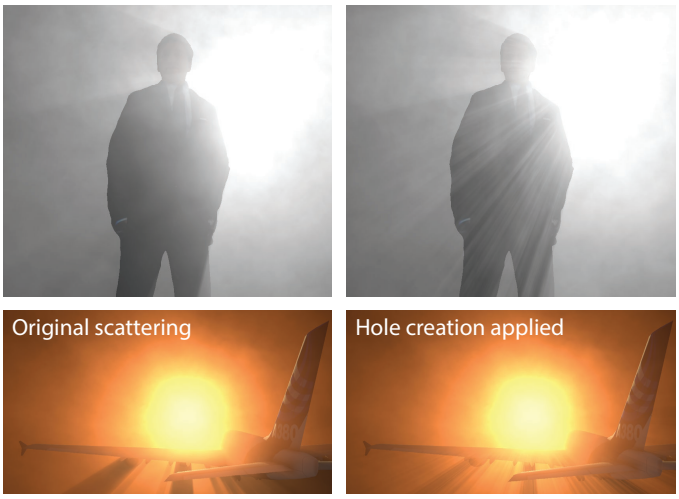

Fig. 9. Hole creation applied to solid objects. Left: the original image. Right: image obtained by creating holes in the shadow map. Top: emphasizing a character. Bottom: simulating motion.

Transfer Functions

Transfer functions are an effective way to influence the light-shaft colors. More specifically, a transfer function (TF) maps the properties of a view ray (effectively a pixel) to the scattering component's output color. In order to make their definition easy to specify for the user, we focus on a mapping of two parameters to a color. Consequently, the transfer function can be defined by a 2D texture, similar to the X-Toon approach [25]. In this way, a TF is also not limited to a single scene; instead, it is possible to transfer the mood caused by a TF to another scene.

As a view ray is uniquely defined by its underlying pixel, an artist can easily influence the entire scene appearance in a consistent and effective way by defining and modifying a transfer function. As an example, an artist might want a certain set of pixels with similar properties to change to an orange color for stylization purposes, which can easily be achieved by a TF. To this extent, the properties along a ray would simply be mapped to the wanted color.

In practice, we use the average visibility along the view ray and the linearized depth of the first surface as parameters for influencing the scattering component's color. Based on these parameters, when using a gradient in the transfer function, the result remains plausible, as the two parameters are often used in realistic approaches as well. Further, please note that these values are along the view ray; i.e., we rely on 3D information, otherwise the stylization would appear to be a 2D overlay, which becomes very apparent when the camera moves. We also experimented with other parameters, such as the average position of visible samples along the view ray, and the angle between the view ray and light direction, but these seem more difficult to use and are therefore not included in our results.

Artists are able to interactively design the TF in our framework to explore various possibilities in real time. The on-the-fly editing of the TF texture uses a painting utility based on layers. In our prototype, solid colors, gradients, images, and special *diffusion layers* are supported (see Figure 10), which can be blended to produce the final TF, using

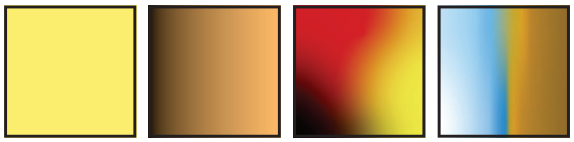

Fig. 10. Different layer types supported by our TF editor: solid, gradient, diffusion, and image layers.

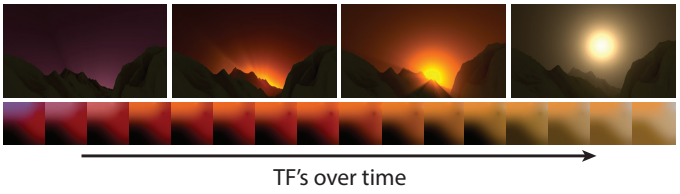

Fig. 11. A sunrise created using several transfer functions. Top: resulting scattering. Bottom: corresponding TFs due to linear interpolation between five user-defined TFs. It can be seen that key-framing the TF easily extends color modifications of light shafts to the time domain.

multiplicative, additive, or alpha blending.

Diffusion layers contain constraints, consisting of a position in the TF texture, a color, and an alpha value. The user can place constraints anywhere in the texture, whose axes represent the parameter domain of the TF. The constraints are diffused throughout the entire layer similar to diffusion curves [33]. A single constraint results in a uniform color, two produce a gradient, and more create complex color combinations, like the third image in Figure 10. Additionally, other diffusion constraints could be integrated [34]. For stacked layers, alpha blending is guided by the alpha values of the constraints. This diffusion throughout the layer creates smooth transitions in parameter space and, thus, all pixels in the scene with similar parameters change similarly. This property makes it easier to produce consistent definitions and renders the tool very effective.

However, defining the TF directly in parameter space can be cumbersome and it is more desirable to define scattering directly at a point in space. For this reason, an artist can simply select a location by clicking on the screen to define a corresponding 3D constraint. The constraint's position in parameter space is computed by querying the pixel's underlying view ray parameters and the user can choose the color constraint to be placed in the TF. Additionally, we allow the recovery of the underlying position in the scene. This position is expressed in barycentric triangle coordinates, which makes it possible to project the point in each frame to the screen and move the constraint accordingly in the parameter space of the TF. To further extend the support of dynamic scenes, we also introduce key-framed transfer functions to produce stylized animations by smoothly interpolating between the TFs defined over time. In this way, expressive scattering can be extended to dynamic scenes, which is especially interesting for predefined scene animations, e.g., an in-game cut-scene with a known camera path. Figure 11 shows this technique to produce a sunrise scene.

Finally, the stylization can be used in combination with a standard scattering model; here, the TF can be monochrome, to serve as a modulator of the scene appearance, while the light color is defined by the source itself. Consequently, surface lighting and scattering remain consistent.

Light Map Optimization

Our approach also offers control over light color and, thus, the color of light shafts for a directional light source. A light map stores the light's color for a given angular direction (in case of a directional source, it can be interpreted as a projection of a texture in the scene). By only influencing the light emission, the results remain physically plausible, which can be a desirable goal in certain situations.

Formally, a light map defines the spatial variation of the unoccluded incoming light $\widetilde{L_\mathrm{i}}(\mathbf{x}_t)$ in Equation 1. At a point $\mathbf{x}$ the light color is determined by projecting the point into light space and performing a texture lookup using its x, y coordinates. As before, we focus on directional light, hence, the light space is defined via an orthographic projection along the light direction. In principle, it could be the same transformation as the shadow map of the light source, but we allow an independent definition to make a differing resolution and focus possible.

We face two main challenges when using such light maps to stylize light shafts. First, there is no intuitive link between individual texels of a light map and the colors perceived after the scattering. Second, we target the creation of a static light map for the entire scene; i.e., for every possible view. Light shafts, however, are highly view-dependent, which in case of static illumination (and thus a view-independent light map), leads to an over-constrained problem. We solve these issues by using an optimization scheme to find a light map, which fulfills the user constraints for a given set of input views as good as possible. In other words, for a given view, the user can draw simple color strokes to express a desired color for the seen light shafts (top row in Figure 17). When all constraints have been collected, we apply an approach similar to the one proposed by Klehm et al. [28] to derive an optimal light map.

Formally, we treat the strokes in the user-defined views as a collection of N *constraint pixels*. Given the corresponding view, a pixel with index $k \in [1, N]$ corresponds to a ray (origin $\mathbf{x}^k$ and direction ω^k). The stroke's color to which the pixel belongs defines the desired color L^k_desired. While we want to enable users to define colors, we do not want them to provide exact scattering results. Instead, we multiply the painted color with the value that would be obtained for a white light map. $L^k := L^k_\mathrm{desired}\, L^\mathrm{white}_\mathrm{scat}(\mathbf{x}^k, \omega^k)$. Hence, the color definition is independent of any shading effects. Our goal is to derive RGB color values $L_\mathrm{e}(s,t)$ for the texels of a light map such that rendering with the light map, yields the expected scattering result $L_\mathrm{scat}(\mathbf{x}^k, \omega^k)$, which should match the target value: $L^k = L_\mathrm{scat}(\mathbf{x}^k, \omega^k)$.

Stacking the texels of the light map in a vector $\vec{x} := (\ldots, L^i_\mathrm{e}, \ldots)^\top$ and the user indications (as defined above) in a vector $\vec{b} := (\ldots, L^k, \ldots)^\top$ defining the target radiance values, we can describe the light transport in form of a linear dependency: $\mathbf{T}\vec{x} = \vec{b}$, where the values of $\mathbf{T}$ are fully described by Equation 1. Unfortunately, it is unlikely that a general solution exists, which would satisfy all equations at once. A simple example is a checkerboard, which cannot be achieved by modifying the light map alone, as light shafts always become darker the farther they are away

from the light source. Hence, we opt for a compromise by finding a solution in the least-squares sense: $\mathbf{T}^\top\mathbf{T}\vec{x} = \mathbf{T}^\top\vec{b}$. Accordingly, the optimal $\vec{x}$ is found by minimizing the quadratic function $f(\vec{x}) = \frac{1}{2}||\mathbf{T}\vec{x} - \vec{b}||^2$ with gradient $\nabla f(\vec{x}) = \mathbf{T}^\top(\mathbf{T}\vec{x} - \vec{b})$.

While any off-the-shelf optimizer can be used, we employ an iterative conjugate gradient method [35] to find the optimal $\vec{x}$. The advantage of a conjugate gradient method is that matrix $\mathbf{T}$ does not need to be stored explicitly and we can easily employ the GPU. We only need to be able to compute the matrix-vector multiplications $\mathbf{T}\vec{x}$ and $\mathbf{T}^\top\vec{b}$; $\mathbf{T}\vec{x}$ corresponds to evaluating the same equations as for rendering with the light map $\vec{x}$, the term $\mathbf{T}^\top\vec{b}$ is an inversion, i.e., pixel values are back-projected onto the light map. While the standard rendering process reads a light map value at each marching step to accumulate the in-scattered light, the back projection to compute $\mathbf{T}^\top\vec{b}$ writes the pixel value $\vec{b}^k$ modulated by the scattering weights to the light map. For the j^{th} marching step with step size t for pixel k, the scattering weight is $\mathbf{T}[k,j] = \mathbf{T}^\top[j,k] = \rho f V(\mathbf{x}_j)\, e^{-jt\sigma_\text{t}}$ (cf. Equation 1) with $\mathbf{x}_j = \mathbf{x}^k + jt\omega^k$. The equation being quadratic, the solution is unique, and we can initialize our light map with a constant.

While the underlying process of the light map optimization is more involved in comparison to the use of transfer functions, this complexity is completely hidden from the user. The approach maintains physical plausibility and is easily controlled via a painting metaphor, in which users can freely decide on the viewpoints and strokes they paint, leading to a spatially varying control.

4.3 Heterogeneity Modification

As illustrated on the right in Figure 2, light shaft irregularity – caused by a heterogeneous medium – plays an important role for stylization. We aim to enable artistic control over the heterogeneity, for which we propose three stylization concepts: first, denoting for locations in the scene to what extent they exhibit heterogeneity; second, controlling the variation and patterns in these irregular areas with locally varying frequencies; and finally, indicating the scattering intensity for scene locations.

In order to apply these concepts, we first need to address the way we will represent heterogeneity. A naive representation would define the medium in a fine 3D grid that contains a spatially varying extinction coefficient. To compute the final scattering, the grid is then sampled using ray marching, simultaneously resolving the visibility (cf. Equation 1). For high-quality light shafts, we would need a substantial amount of sample points to avoid discretization artifacts in the visibility sampling, which would yield low performance, but efficient solutions for heterogeneous media are currently lacking. Instead, we make use of two approximations. First, we apply heterogeneity in a post-process by modulating the homogeneous result, which delivers a pleasing appearance. Second, we observe that heterogeneity in thin media does not lead to strong high-frequency changes, which allows us to coarsely sample this information, making it possible to use a coarser grid.

In practice, we consider a 256^3 grid, the *noise volume*, filled with 3D Perlin noise values between 0 and 1, like its 2D counterpart described in Section 4.1. Note the shadow map resolution is typically much higher. To compute a pixel's final color, we multiply the scattering result L_scat (see Equation 2) by the average heterogeneity $h(\mathbf{x},\omega_\text{i})$ for this view ray, obtained from ray marching through the noise volume (Figure 12):

Fig. 12. Comparison between using a homogeneous medium (left) and our heterogeneity approximation (right).

$$\hat{L}_\text{scat}(\mathbf{x},\omega_\text{i}) = L_\text{scat}(\mathbf{x},\omega_\text{i})\; h(\mathbf{x},\omega_\text{i})$$
$$h(\mathbf{x},\omega_\text{i}) := s^{-1}\int_0^s N(\mathbf{x}_t)\; dt.$$

To stylize this result, we define three *blending volumes*, one controls the amount of heterogeneity, the next the noise frequency, and the final the scattering intensity:

$$h(\mathbf{x},\omega_\text{i}) := \int_0^s I(\mathbf{x}_t)\; (\; \alpha(\mathbf{x}_t)N(\mathbf{x}_t, f(\mathbf{x}_t)) + (1-\alpha(\mathbf{x}_t))\;)\; dt.$$

The heterogeneity blending volume defines the linear interpolation α between the values obtained from the noise volume and a constant homogeneity. The noise frequency blending volume influences the noise frequency by defining f, the mipmap level used to look up the noise. Hereby, local noise modifications become possible, while the actual Perlin noise parameters can be used to define a global control of variations and patterns in the heterogeneity. Finally, the intensity blending volume defines I, which scales the overall scattering contribution.

The blending weight volumes are modified via a 3D painting tool consisting of a sliding plane orthogonal to the camera direction, which the user can move along its normal. An ellipsoidal brush and eraser, with a user-controlled size, opacity and hardness, are available to draw on this plane, which produces a 3D splat in the selected blend volume. Note that the noise and blending volumes can be baked into one volume to make rendering more efficient.

5 Results and Discussion

To assess our methods, we present our stylization results below. As our methods are executed on the fly, we also discuss performance and the applied optimizations.

5.1 Stylization

Occluder Manipulation

Figure 13 demonstrates occluder manipulations. The palm tree scene exhibits many small light shafts that are especially noticeable under camera animation (see video). Using our closure operation to fill holes in the shadow map, we can effectively remove small light shafts. Note that the 2D filter can cause artificial occlusions in mid-air due to the extension of the occluders into empty space, which can produce subtle

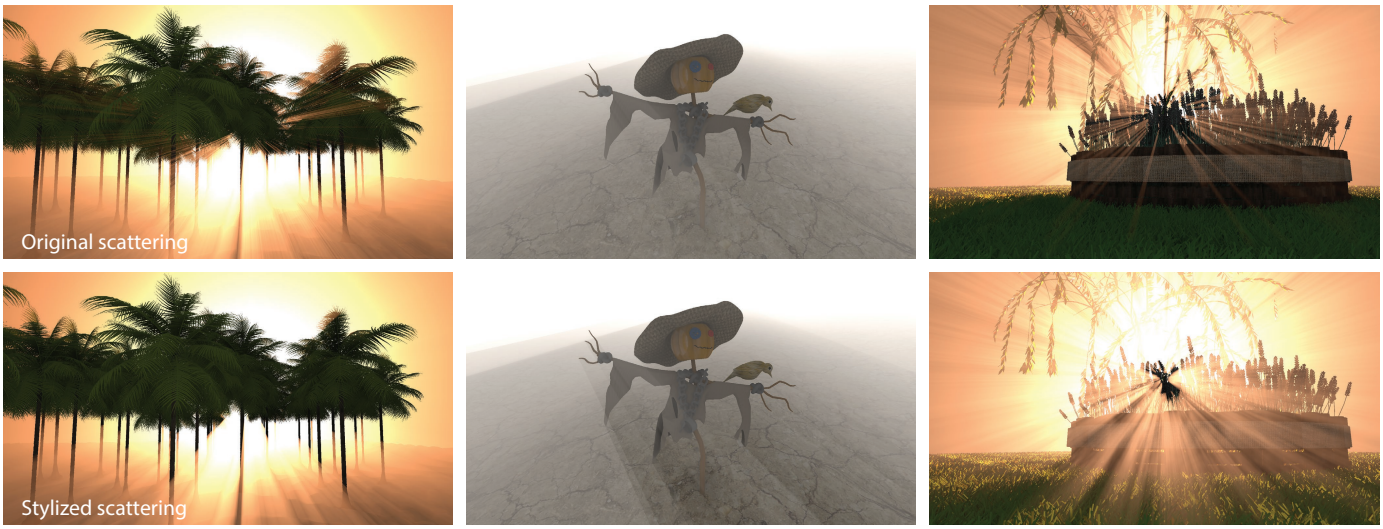

Fig. 13. Stylized scattering applied on the palm tree, scarecrow and flowerbed scenes using our occluder manipulation tools. Top: original images with physically-based scattering. Bottom: resulting images. From left to right: hole filling, silhouette enhancement and hole creation.

dark halos around objects. Alternatively, one could use a 1D epipolar closure, which forces the occlusion to be behind visible scene objects. Nonetheless, in practice, this artifact goes usually unnoticed (as in the palm tree scene).

The scarecrow scene (Figure 13, center) demonstrates the enhancement of light shaft edges using a 1D epipolar dilation. Here, the effect is used to increase a feeling of fear and leads to an emphasis of the object. A similar harshness is often used in comics to illustrate activity and stress the importance of an object. As the technique practically removes some of the scattering, it may darken the image slightly, which can be compensated for using a transfer function. Such a combination is shown in Figure 16.

Finally, hole creation enables artists to add light shafts, as illustrated in the flowerbed scene (Figure 13, right). Objects that initially exhibit a set of complex light shafts can profit from this operation.As discussed in Section 4.1, the Perlin noise parameters can be modified to match the intended appearance. Figure 14 shows the effect of different parameters on the teaser's example scene. Small holes (bottom) can be reduced in size until they average out due to the integration along the view ray, while large holes (top) can drastically simplify the scattering appearance. The control supplied by these parameters enables transitioning between physically plausible scattering with the original occluders, and the exaggerated alternatives. The complex geometry of the tree makes the hole creation process appear very natural. In general, we expect hole creation to be mostly applied in similar situations.

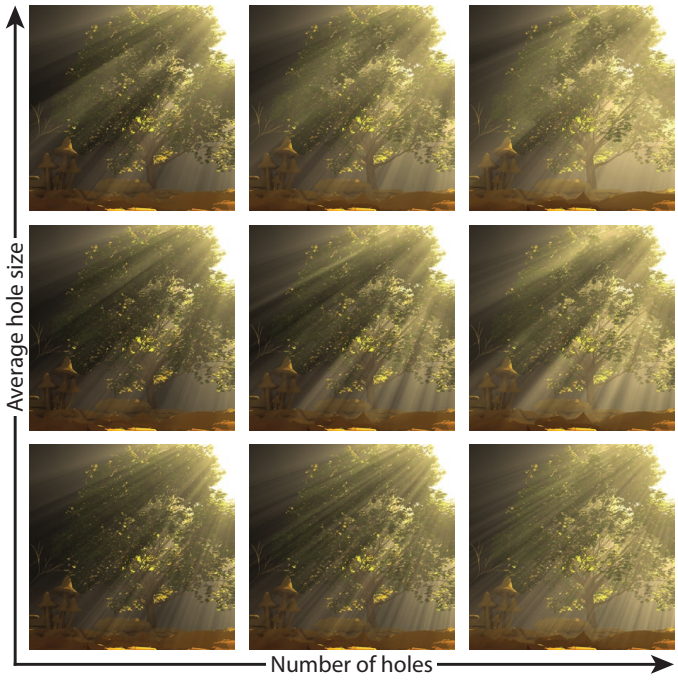

Fig. 14. Effect of Perlin noise parameters on the resulting scattering.

Color Modifications

Transfer functions make it possible to achieve quick and impressive changes in the overall appearance of the scattering process by defining colors independently of surface, light, or medium parameters. These modifications enable artists to quickly redefine the mood of a scene, as illustrated in Figure 15. Here, the TF creates an alarming atmosphere in the scenes by adding multiple colors with strong edges between them leading to quantization effects in the resulting scattering. Figure 15 also illustrates the possibility to transfer a given style from one scene to the next; the terrain's TF can directly be used to stylize two more scenes. Additionally, the city and the playground scenes are both examples of how stylized scattering can be used to emphasize an object.

In general, all our techniques can be used in combination to achieve a complex interplay. Occluder manipulation coupled with TFs is shown in the turtle scene (Figure 16). The initial shot exhibits an unlucky overlap of occluders (head, body and hind fins). An artist may want to edit the scattering to put more emphasis on the actual object and remove attention from the light shafts. The 2D morphological filtering closes the hole between the hind fins, which gives the light shafts a more simplified appearance. Then, a

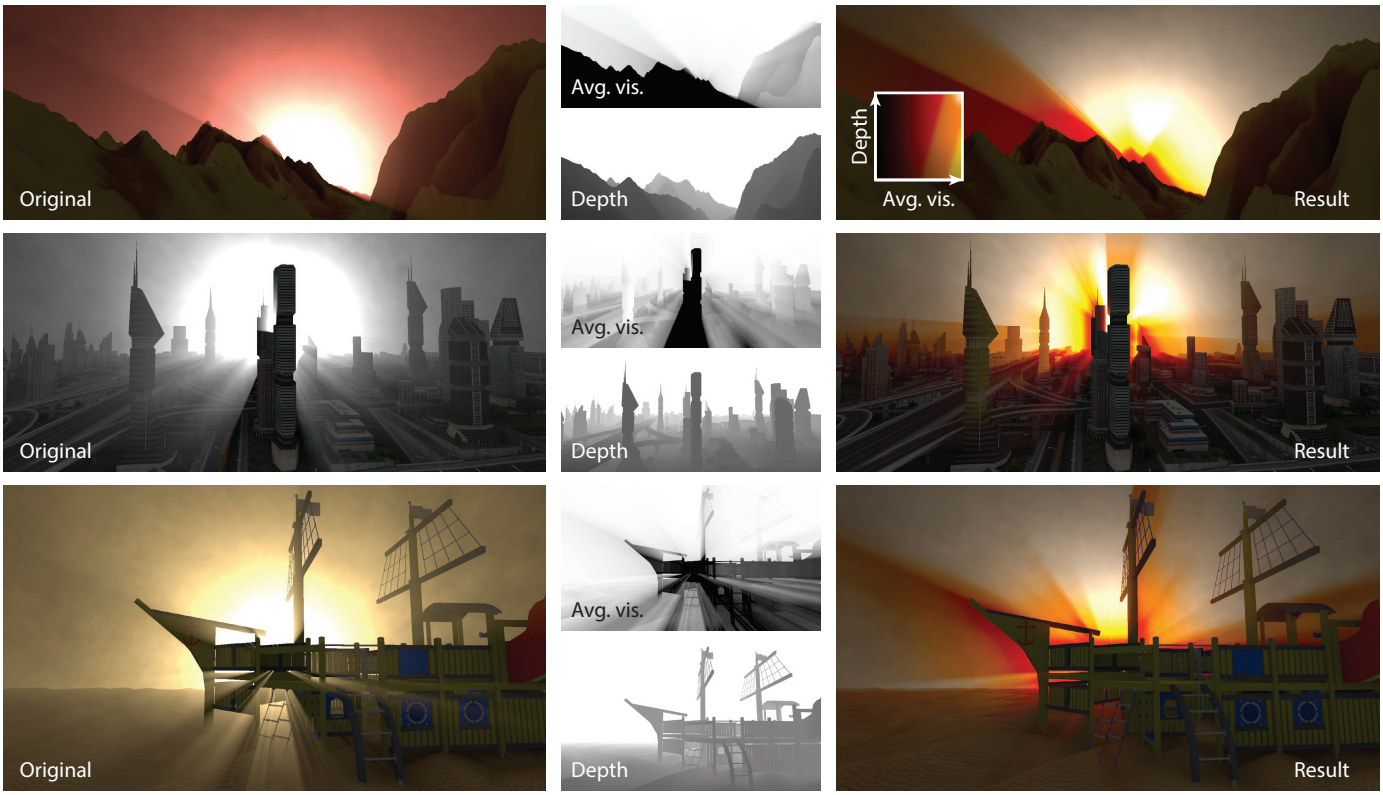

Fig. 15. Expressiveness of transfer functions. Physically-based scattering (left) is stylized (right) using a transfer function (inset), which is parametrized by the average visibility and linearized depth of the view rays (center).

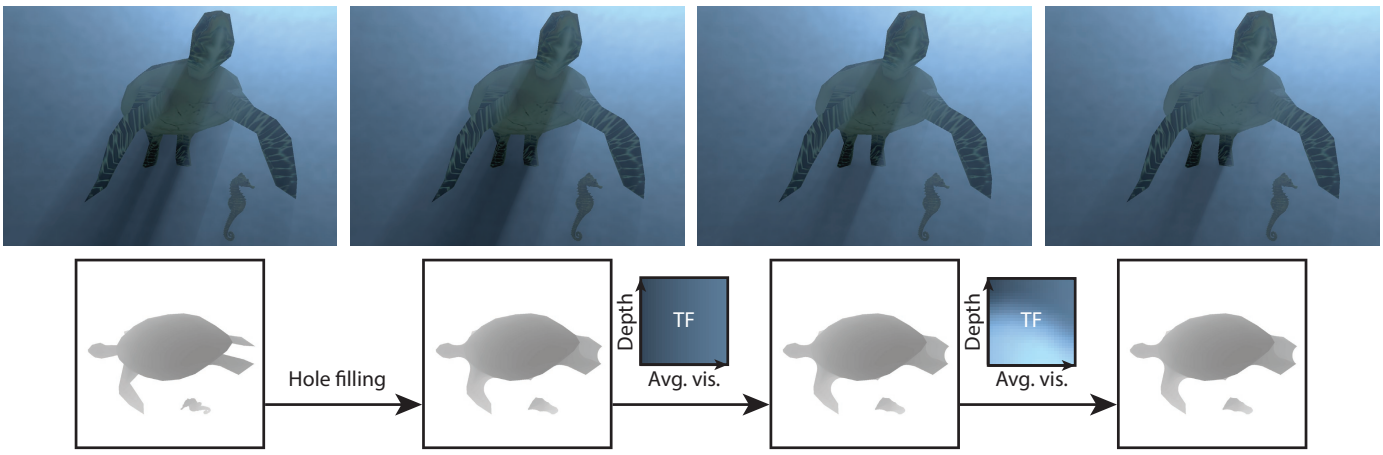

Fig. 16. Combined use of occluder manipulation and TFs. From left to right: the original image; hole filling unifies the hind fins; a simple TF is applied to reduce the darkness of the shadow; another TF is used to remove the dark patch on the turtle's body.

TF enables reducing the shadows, while keeping a plausible scattering appearance. Taking it further, we can use a TF to remove the potentially distracting dark stripe on the body caused by the shadow of the head, as the pixels in this area have similar average visibility and scene depth.

Figure 17 shows an example of light shaft coloring via our light map optimization. Here, colors are drawn by a user for three different views of the citadel scene. Based on this input, we generate a light map (bottom center). Using it for the scattering computation gives us the results shown in the middle row, which exhibit colored light shafts matching the user-defined indications. Note that our optimization scheme ensures that the light shafts are colored as specified by the user, despite a potential overlap. Some red and white light also becomes visible in the right image, where the user did not specify any additional constraints. If this result is undesired, it could be rectified by additional edits.

Transfer functions and light map optimization are complementary techniques for light shaft colorization; while the former is better suited for controlling the general mood of a scene, and can even be transferred to other scenes, the latter is of more use in specific scenarios and eases local changes.

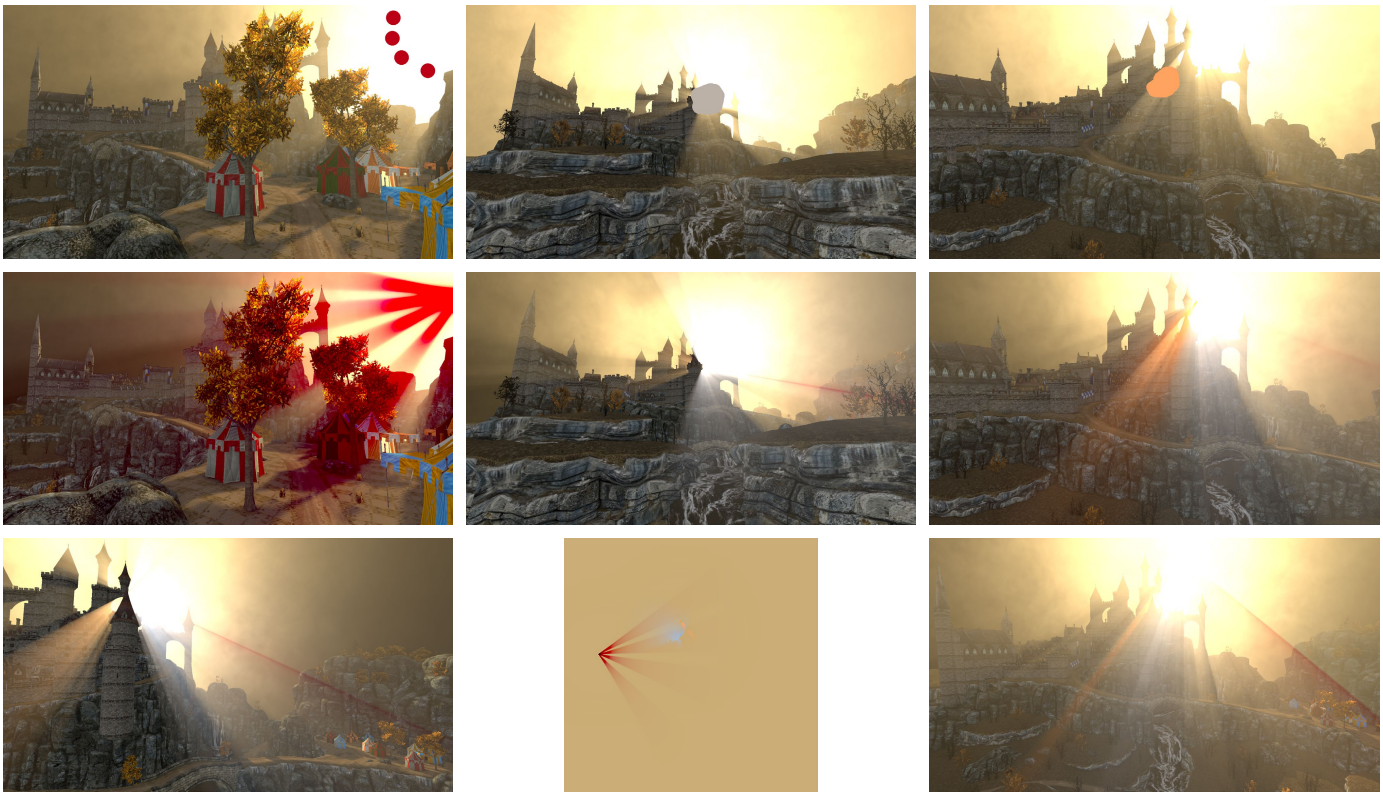

Fig. 17. Coloring light shafts with light map optimization. Top: three views in which a user has drawn colors. Middle row: resulting images, which display light shafts that correspond to the specified colors. Bottom: two intermediate views are shown, which demonstrate a smooth blending and fade-out of the colors. The light map that was generated with this user input is shown in the center.

Fig. 18. Heterogeneity modification. From left to right: the original scattering produces regular light shafts; heterogeneity is approximated using a 3D Perlin noise texture, which creates more irregular scattering effects and gives the medium a distinguishable volumetric structure; using our 3D painting tool, we can remove unwanted details that are due to heterogeneity while retaining the irregularity of the light shafts.

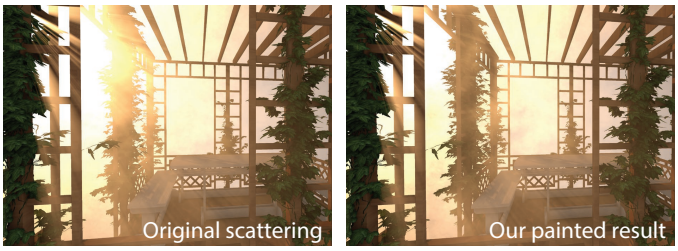

Fig. 19. The scattering intensity can also be controlled using our 3D painting tool. Here, the unwanted saturation in the upper left corner is removed to reveal more of the vines' details.

Heterogeneity Modification

Our heterogeneity approach is demonstrated for the dinosaur scene in Figure 18. Here, we define the trade-off between heterogeneity and homogeneity directly via a painting approach in 3D. First, the background in the middle image was chosen and its homogeneity increased to avoid distracting details in the medium (in the upper part of the image), which would have taken emphasis from the dinosaur skeletons. Note that the irregularity of the light shafts in the lower part of the image is maintained.

Modulating the scattering intensity can also find useful applications. Such a change is illustrated for the arbor scene (Figure 19), where overexposed areas are toned down.

To produce a result in the spirit of the right image in Figure 2, we can combine modifications of all three blend volumes (Figure 20). Here, we drastically increase the scattering intensity between the houses to simulate fog, as well as for the light shafts caused by the opening of the citadel wall – the latter are also modified to be fully homogeneous to avoid distracting details. Furthermore, we use a higher noise frequency for parts of the background

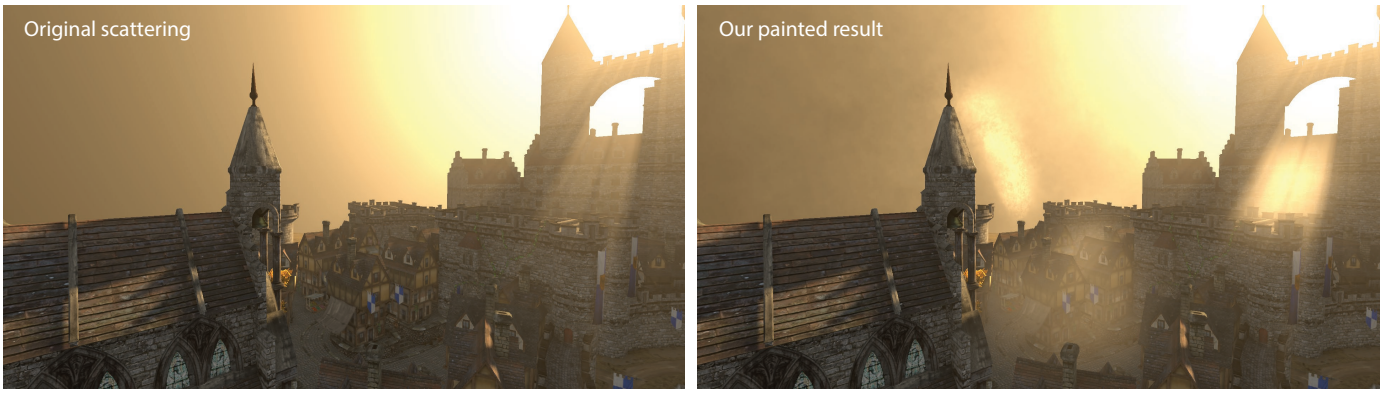

Fig. 20. Combination of painting in all three blend volumes. Left: original image using a homogeneous medium. Right: the painted result.

TABLE 1
Performance (in ms) of our prototype for the tree scene in Figure 1, rendered in Full HD for different shadow map (SM) sizes. Measurements for hole creation include Perlin noise creation and SM modification. A transfer function (TF) for color stylization is indexed using the depth map of a G-buffer [36] as well as the view ray average visibility. Using an acceleration method like the one from Klehm et al. [2] is required as naive ray marching is an order of magnitude slower (67.8ms for Full HD and 1024 marching steps).

SM Size	SM Creation	Holes	G-buffer	Visibility	TF
512^2	1.3	0.1	4.0	1.6	0.2
1024^2	1.6	0.3	3.9	2.5	0.2
2048^2	4.0	1.4	4.1	4.6	0.2

TABLE 2
Performance (in ms) for hole filling for the tree scene of Figure 1, rendered in Full HD for different SM and kernel sizes. As we work in image space, the kernel size needs to be adapted to the SM size as well as the content. The right-most column gives an indication of how many holes are filled for the given SM and kernel size.

SM Size	2D Closure		1D Epipolar Closure		Filled
	Kernel	Filtering	Kernel	Filtering	
512^2	11	1.1	15	0.6	All
512^2	21	3.7	30	1.1	All
512^2	41	13.3	60	1.9	All
1024^2	11	4.5	15	1.7	Most
1024^2	21	15.4	30	2.9	All
1024^2	41	56.7	60	5.2	All
2048^2	11	18.0	15	5.4	Many
2048^2	21	60.9	30	9.3	Most
2048^2	41	222	60	16.8	All

to simulate chimney smoke. Additional examples of all our stylization techniques and animated results are shown in the supplemental video.

5.2 Performance

Occluder manipulation and transfer functions work in image space, with the former directly modifying the shadow map that is used for the scattering computation. Image space has several benefits over object space. The performance does not depend on geometric detail or scene complexity and the methodology is well-suited for today's parallel hardware. Furthermore, operations such as hole filling are easy to perform, because the neighborhood of objects is automatically resolved. Tables 1 and 2 show performance measurements on an NVIDIA Titan in Full HD for occluder manipulation and pipeline steps for computing and coloring light shafts.

As we have shown, computing single scattering in homogeneous media is possible at high frame rates using specialized and efficient techniques that circumvent naive ray marching. However, to approximate heterogeneity, we need ray marching to evaluate the spatially varying scattering intensity along the view ray. When directly applying a marching process, consisting of several thousand steps per pixel, it becomes the bottleneck in our framework. As discussed in Section 4.3, we mitigate this problem by outfactoring the intensity, which allows us to cut down the number of marching steps. Figure 21 shows results for multiple numbers of marching steps.

While 2048 marching steps do not create improved results in a 256^3 volume, it indicates the high costs that would

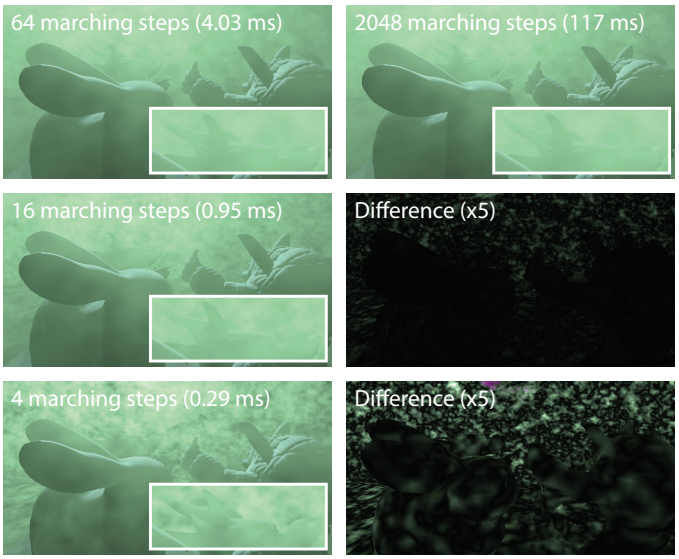

Fig. 21. Effect of marching steps using our heterogeneity approximation. Timings obtained in Full HD. Top: result with 64 steps and reference image with 2048 marching steps (no difference image because it is identical to the reference). Middle: result with 16 steps and the corresponding difference to the reference, multiplied by 5 for illustration. Bottom: result with 4 steps and the corresponding difference image.

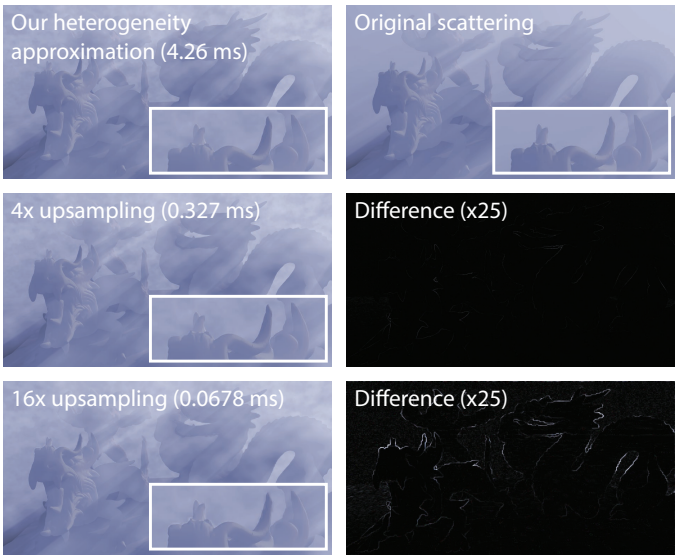

Fig. 22. Effect of 2D upsampling using our heterogeneity approximation. Timings obtained in Full HD. Top: reference image without upsampling and the original scattering. Middle: 4x upsampling result, and the corresponding difference to the reference, multiplied by 25 for illustration. Bottom: 16x upsampling result, and the corresponding difference image.

arise if not relying on any approximation. Even at 16 steps, our approximation results in a reasonable visual quality. Only at 4 steps, the results are no longer acceptable. In practice, we use 64 steps, which is 29 times faster compared to 2048 steps, and easily reaches real-time performance.

An additional acceleration is possible without a significant quality loss by using an depth-map-driven upsampling [37]. The approach works well because heterogeneity produces smooth effects. Figure 22 shows 4x and 16x upsampling using 64 marching steps. For 4x upsampling the difference is negligible but uses 16 times less pixels. Visible artifacts only occur for higher upsampling rates, such as 16x (reduction by a factor of 256). The speed-up is 13 times using 4x upsampling, leading to real-time rates without discernible quality loss.

The light map optimization in Figure 17 computes in 2.04 seconds, quick enough to allow for interactive editing. Here, we painted a total of 41240 constraint pixels from 3 different views. The resolution of the light map was set to 512×512 with a corresponding shadow map of size 512×512 and a ray marching procedure with 768 steps. The optimization converges after 32 conjugate gradient iterations.

5.3 Discussion

In our results, we use directional light only, as light shafts are generally caused by the sun, both in virtual scenes (see Figure 2) and real life, for which directional light is a good approximation. However, the methods presented can be extended to support non-directional light as well, as they pose no additional requirements on the light source besides the presence of a shadow map. Furthermore, while we only support single scattering, this suffices for optically thin media where light shafts occur. Additionally, we can approximate a heterogeneous medium, enabling further applications of our techniques. In summary, our methods cover all practical and important use cases of light shafts.

To assess the practical usefulness of our methods, we consulted two professional production artists working in the visual effects industry. They said light shafts are very common in production, and used as a stylistic tool or mood changer, for which artistic control is essential. However, only limited and laborious approaches exist to control light shafts, such as placing large proxy objects, or entirely faking them in post-processing. For this reason, our tool was enthusiastically received, and considered to be easy to use with a good level of parameter control. Requests were even made to make a product out of it. For more details, we refer to the supplemental document.

6 Conclusion

We have presented several strategies to stylize volumetric single scattering, overcoming the difficulty that light shafts depend on the layout of an entire environment. Our approach is compatible with animated scenes and relies on very efficient solutions, which makes it ready to be used for real-time applications, and enables a quick exploration of the various settings. The techniques are applied at a global scope – i.e., for the whole scene – but can also be used to make local changes to the scattering behavior. Image-based occluder manipulations modify the complexity of the scattering appearance and are controlled by only a few parameters. Transfer functions allow us to interactively design a general mood and the result can even be transferred to other scenes. As an alternative, users can design a light map to modify the light emittance by relying on an optimization process which ensures that user-defined constraints are respected, which are defined using a painting metaphor. Furthermore, we employ an efficient algorithm to approximate heterogeneity and enable the control of scattering intensity, noise frequency, and heterogeneity ratio. Finally, our solution supports key-framed animation to steer the stylization over time.

Our system makes a step towards designing scattering behavior, but leaves room for future work via additional object-focus strategies, stylization techniques for multiple scattering, or more advanced transfer function parameters.

Acknowledgments

This work was partly supported by the Intel Visual Computing Institute at Saarland University.

References

[1] J. Chen, I. Baran, F. Durand, and W. Jarosz, "Real-time volumetric shadows using 1d min-max mipmaps," in *i3D*, 2011, pp. 39–46.
[2] O. Klehm, H.-P. Seidel, and E. Eisemann, "Prefiltered single scattering," in *i3D*, 2014, pp. 71–78.
[3] D. Nowrouzezahrai, J. Johnson, A. Selle, D. Lacewell, M. Kaschalk, and W. Jarosz, "A programmable system for artistic volumetric lighting," *ACM TOG (SIGGRAPH)*, vol. 30, no. 4, pp. 29:1–29:8, 2011.
[4] O. Klehm, I. Ihrke, H.-P. Seidel, and E. Eisemann, "Volume stylizer: Tomography-based volume painting," in *i3D*, 2013, pp. 161–168.
[5] O. Klehm, T. R. Kol, H.-P. Seidel, and E. Eisemann, "Stylized scattering via transfer functions and occluder manipulation," in *GI*, 2015, pp. 115–121.
[6] F. Pellacini, K. Vidimče, A. Lefohn, A. Mohr, M. Leone, and J. Warren, "Lpics: A hybrid hardware-accelerated relighting engine for computer cinematography," *ACM TOG (SIGGRAPH)*, vol. 24, no. 3, pp. 464–470, 2005.

[7] J. Ragan-Kelley, C. Kilpatrick, B. W. Smith, D. Epps, P. Green, C. Hery, and F. Durand, "The lightspeed automatic interactive lighting preview system," *ACM TOG (SIGGRAPH)*, vol. 26, no. 3, pp. 25:1–25:11, 2007.
[8] J. Obert, J. Křivánek, F. Pellacini, D. Sykora, and S. Pattanaik, "icheat: A representation for artistic control of indirect cinematic lighting," *Comp. Graph. Forum (EGSR)*, vol. 27, no. 4, pp. 1217–1223, 2008.
[9] W. B. Kerr, F. Pellacini, and J. D. Denning, "Bendylights: Artistic control of direct illumination by curving light rays," *Comp. Graph. Forum (EGSR)*, vol. 29, no. 4, pp. 1451–1459, 2010.
[10] Y. Song, X. Tong, F. Pellacini, and P. Peers, "Subedit: A representation for editing measured heterogeneous subsurface scattering," *ACM TOG (SIGGRAPH)*, vol. 28, no. 3, p. 31, 2009.
[11] O. Mattausch, T. Igarashi, and M. Wimmer, "Freeform shadow boundary editing," *Comp. Graph. Forum (EG)*, vol. 32, pp. 175–184, 2013.
[12] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz, "Stylized shadows," in *NPAR*, 2007, pp. 77–83.
[13] M. Ament, F. Sadlo, C. Dachsbacher, and D. Weiskopf, "Low-pass filtered volumetric shadows," *IEEE TVCG*, vol. 20, no. 12, pp. 2437–2446, 2014.
[14] D. Patel, V. Šoltészová, J. M. Nordbotten, and S. Bruckner, "Instant convolution shadows for volumetric detail mapping," *ACM TOG*, vol. 32, no. 5, pp. 154:1–154:18, 2013.
[15] D. Gutierrez, F. J. Seron, J. Lopez-Moreno, M. P. Sanchez, J. Fandos, and E. Reinhard, "Depicting procedural caustics in single images," *ACM TOG (SIGGRAPH Asia)*, vol. 27, no. 5, pp. 120:1–120:9, 2008.
[16] J. Schmid, R. W. Sumner, H. Bowles, and M. Gross, "Programmable motion effects," *ACM TOG (SIGGRAPH)*, vol. 29, no. 4, pp. 57:1–57:9, 2010.
[17] S. Lee, E. Eisemann, and H.-P. Seidel, "Depth-of-field rendering with multiview synthesis," *ACM TOG (SIGGRAPH Asia)*, vol. 28, no. 5, pp. 134:1–134:6, 2009.
[18] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella, "Directing gaze in 3d models with stylized focus," in *EGSR*, 2006, pp. 377–387.
[19] T.-W. Schmidt, F. Pellacini, D. Nowrouzezahrai, W. Jarosz, and C. Dachsbacher, "State of the art in artistic editing of appearance, lighting, and material," in *Eurographics State of the Art Reports*. Eurographics Association, 2014.
[20] M. Hašan and R. Ramamoorthi, "Interactive albedo editing in path-traced volumetric materials," *ACM TOG (SIGGRAPH)*, vol. 32, no. 2, pp. 11:1–11:11, 2013.
[21] C. Wyman, "Voxelized shadow volumes," in *HPG*, 2011, pp. 33–40.
[22] M. Billeter, E. Sintorn, and U. Assarsson, "Real time volumetric shadows using polygonal light volumes," in *HPG*, 2010, pp. 39–45.
[23] O. Elek, T. Ritschel, C. Dachsbacher, and H.-P. Seidel, "Interactive light scattering with principal-ordinate propagation," in *GI*, 2014, pp. 87–94.
[24] T.-W. Schmidt, J. Novák, J. Meng, A. S. Kaplanyan, T. Reiner, D. Nowrouzezahrai, and C. Dachsbacher, "Path-space manipulation of physically-based light transport," *ACM TOG (SIGGRAPH)*, vol. 32, no. 4, p. 129, 2013.
[25] P. Barla, J. Thollot, and L. Markosian, "X-toon: An extended toon shader," in *NPAR*, 2006, pp. 127–132.
[26] H. Guo, N. Mao, and X. Yuan, "WYSIWYG (what you see is what you get) volume visualization," *IEEE TVCG*, vol. 17, no. 12, pp. 2106–2114, 2011.
[27] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg, "Painting with light," in *SIGGRAPH*. ACM, 1993, pp. 143–146.
[28] O. Klehm, I. Ihrke, H.-P. Seidel, and E. Eisemann, "Property and lighting manipulations for static volume stylization using a painting metaphor," *IEEE TVCG*, vol. 20, no. 7, pp. 983–995, 2014.
[29] K. Zhou, Q. Hou, M. Gong, J. Snyder, B. Guo, and H.-Y. Shum, "Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media," in *IEEE Computer Graphics and Applications*. IEEE, 2007, pp. 116–125.
[30] I. Baran, J. Chen, J. Ragan-Kelley, F. Durand, and J. Lehtinen, "A hierarchical volumetric shadow algorithm for single scattering," *ACM TOG (SIGGRAPH Asia)*, vol. 29, no. 6, pp. 178:1–178:10, 2010.
[31] V. Pegoraro and S. G. Parker, "An analytical solution to single scattering in homogeneous participating media," *Comp. Graph. Forum (EG)*, vol. 28, no. 2, pp. 329–335, 2009.
[32] K. Perlin, "An image synthesizer," *Computer Graphics (SIGGRAPH)*, vol. 19, no. 3, pp. 287–296, 1985.
[33] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, "Diffusion curves: A vector representation for smooth-shaded images," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, vol. 27, 2008.
[34] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot, "Diffusion constraints for vector graphics," in *NPAR*, 2010, pp. 35–42.
[35] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon University, Tech. Rep., 1994.
[36] T. Saito and T. Takahashi, "Comprehensible rendering of 3-d shapes," *Computer Graphics (SIGGRAPH)*, vol. 24, no. 4, pp. 197–206, Sep. 1990.
[37] L. Yang, P. V. Sander, and J. Lawrence, "Geometry-Aware Framebuffer Level of Detail," *Computer Graphics Forum*, 2008.

Timothy R. Kol is a PhD candidate at Delft University of Technology in the Computer Graphics and Visualization group. He received his master's degree in Game and Media Technology from Utrecht University in 2014 and his bachelor's degree in Computer Science from Delft University of Technology in 2011. His research interests include visualization, real-time rendering and participating media.

Oliver Klehm is about to finish his PhD at the computer graphics group of the Max Planck Institute for Informatics. He recently joined Double Negative Visual Effects, Inc. in London as a Research and Development Programmer, working on offline rendering. His research interests include interactive and offline light transport algorithms, as well as applications to stylization to enable artists to create visually appealing results that support storytelling. He holds a MSc (2011) and BSc (2008) from the Hasso-Plattner-Institut.

Hans-Peter Seidel is a scientific director of the Max Planck Institute for Informatics, chair of the computer graphics group and professor at Saarland University. He is co-chair of the Max Planck Center for Visual Computing and Communication, scientific coordinator of the Cluster of Excellence on Multimodal Computing and Interaction, and a member of the Governance Board of the Intel Visual Computing Institute. He received the DFG Leibnitz-Prize (2003) and the Eurographics Distinguished Career Award (2012).

Elmar Eisemann is a professor at Delft University of Technology, heading the Computer Graphics and Visualization group. Before, he was an associate professor at Télécom ParisTech and senior researcher in the Cluster of Excellence at MPII/Saarland University. His interests include real-time and perceptual rendering, alternative representations, shadow algorithms, global illumination, and GPU acceleration techniques. In 2011, he was honored with the Eurographics Young Researcher Award.