Stylized Scattering via Transfer Functions and Occluder Manipulation

Oliver Klehm[1,2*] Timothy R. Kol[2] Hans-Peter Seidel[1] Elmar Eisemann[2]

[1]MPI Informatik [2]Delft University of Technology

Figure 1: Left: single scattering using the original geometry of the scene. The leaves of the tree block most of the light, causing only a subtle scattering effect. Right: scattering created by occluder manipulation. Using our system, an artist can easily add holes into the shadow map of the tree, causing an increased amount of and more interesting scattering effects. While physically incorrect, it is not visible to the viewer that the right image uses fake occlusion information. Insets show the scattering only. Surface shadows are created from the unmodified shadow map.

Abstract

Volumetric light scattering is an effect that is used increasingly in feature movies as well as games. It enables rendering scenes more realistically and is often used as an artistic tool to achieve a certain mood in the scene or emphasize certain objects. Thus far, however, little research has focused on artistically influencing the air-light integral and scattering process, which are both very complex. We propose a novel solution to help artists in changing the appearance of single scattering effects. Our approach offers techniques based on occluder manipulation to remove or add apparent complexity to the resulting light shafts and to emphasize the object's shape by enhancing the light shaft borders. Furthermore, we adapt an existing shading technique to control the effect of the light integral intuitively through the use of easily modifiable transfer functions. Our solution is easy to use, is compatible with standard rendering pipelines, and can be executed interactively in real time to provide the artist with quick feedback.

Keywords: interactive stylization, artist control, single scattering

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

1 Introduction

Participating media can influence the look of a scene drastically. The appearance resulting from light scattering in these media often adds realism and spatial cues for a better scene understanding, or can serve for artistic purposes. One of the dominant elements resulting from scattering are crepuscular rays (or so-called god rays). These light shafts are caused by light rays that illuminate optically thin participating media, while neighboring rays are blocked. The effect can occur, for example, if there is an opening in the clouds. While visually pleasing, the underlying physical processes is often rather complex and the resulting appearance can be hard to predict. Currently, very few algorithms exist to influence the appearance of the scattering process in an artistic manner. [16, 12]

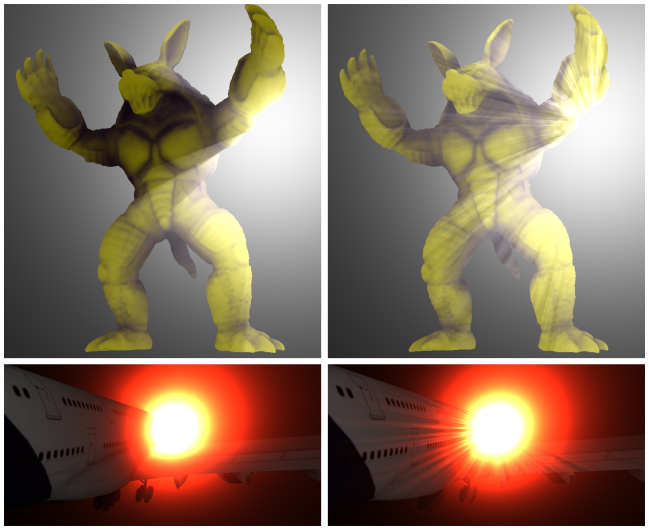

Figure 2: Hole creation applied to solid objects. Left: the original image. Right: image obtained by creating holes in the shadow map. Top row: the creation of a halo around a character. Bottom row: the illusion of motion.

There are several examples that inspired our work. One of these is the whale sequence in the Disney film *Fantasia 2000*, which contains physically implausible light shafts. Also, in comics and animation movies, it is not uncommon to add exaggerated light shafts

*e-mail: {oklehm,hpseidel}@mpi-inf.mpg.de, {t.r.kol,e.eisemann}@tudelft.nl

to characters. The harshness of the lighting can be particularly striking, as most real-world light shaft boundaries appear smooth.

Another example consists in abstractions of the shape. Consider a focus object that is illuminated by a directional light such as the sun. The light shafts are often kept very simple, e.g., even if foliage of surrounding trees would result in a complex pattern, in many cases a simpler representation is used. Inversely, as light shafts can be visually pleasing, they are often exaggerated.

Our work offers new ways of influencing the appearance of light scattering to achieve similar effects by employing two manipulation concepts. First, by changing the occluders in the scene, our algorithm is able to add, remove, and sharpen light shafts. Fig. 1 illustrates an addition of light shafts to brighten the scene and add more detail, leading to a significant change in appearance. Second, our algorithm provides simple controls to achieve various styles and expressive changes through the use of transfer functions, which are easily modifiable to influence the mood for a scene.

As with most artistic tools, it is crucial that users can change the parameters interactively to explore possible choices. For this reason, we focus on stylization and scattering methods that can be computed in real time, and enable immediate feedback when parameters are changed. Our approach is able to consistently stylize the scattering effects in an entire 3D scene, making it suitable for animations. Furthermore, its high-level definitions help in generalizing the settings and the transfer of a general style to various scenes of different geometry, camera, and light settings. Also, we enable artists to create very specific styles for controlled animated scenes, such as cut-scenes in games. Results for these methods are showcased in our supplemental video.

Specifically, our work makes the following contributions:

- light shaft addition, removal, and enhancement using image-based occluder manipulation;
- light shaft color modification via a user-editable transfer function based on view ray properties;
- light shaft animation by dynamic occluder manipulation, and key-framed transfer functions for animated scenes.

2 Related Work

Previsualization systems [20, 22] generally give 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 physically-based, and the possibilities for abstraction and stylization are mostly restricted to the scene setup. In recent years, solutions to influence physics for the purpose of expressiveness have received increasing attention, enabling artists with more possibilities.

There are many approaches to stylize natural phenomena and give control over the appearance of a scene, a few of which we will mention here. Modifications of the light transport [17, 11, 26], shadows [15, 8, 1, 18], motion blur [24], or depth of field [14], have been proposed for significantly influencing the appearance of a scene, as well as for guiding the observer to specific regions of interest. Similar abstraction mechanisms have been demonstrated to be very efficient for focus control [7].

Occluder manipulations as one of the possible modifications have been applied before in the form of proxy geometries to modify light transport in a scene. Schmidt et al. [25] introduced the idea of path-proxy linking, where they define invisible copies of scene objects, which are then 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 interesting for stylizing scattering. To this end, we manipulate occluders using morphological operators, as recently applied by Calderon and Boubekeur to 3D point clouds [5]. However, we work in the space of a 2D shadow map as this gives direct control of creating, removing, and enhancing light shafts.

A change in color is often achieved by the use of transfer functions, such as for surface shading [3] or volume visualization [10]. Unlike in volume visualization [10], 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.

Regarding scattering, only a few approaches have been suggested. Artistic beams [16] let the user modify individual light beams by influencing shape, fall-off, and color. The color definition is optimized to find a plausible mapping to properties of the participating medium. Treating beams individually can be an advantage, but global control becomes more time-consuming or difficult. In contrast, our stylization method uses parameters derived from scattering to directly map to the final scattering color.

One can also rely on a set of painted input images to find the optimal volume parameters to best match the provided target images [12]. In this case, the volume parameters are stored in a voxel grid, which limits the possible resolution and results in computation times far from real time, despite the employment of an efficient process. Furthermore, defining the input requires a certain artistic skill, and the final rendering is bound to the actual physical process, limiting the potential expressiveness.

In our solution, we want to make it easy to define plausible results, but also enable more expressive solutions, potentially decoupled from the physical behavior. For this, efficient computation is key to allow artists to explore various options rapidly. Hence, we focus on existing real-time solutions for single scattering.

There are several options for an efficient single scattering computation; min-max mipmaps [6], voxelized shadow volumes [27], shadow volumes based on shadow maps [4], or prefiltered single scattering [13]. While approaches for multiple scattering exist [9], their precision is still relatively low due to the use of a coarse grid, which is why we concentrate on single scattering only.

3 Background Real-time Scattering

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.

A common approximation is to factor out visibility [2, 6, 27, 13], which we will 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 [19] for a variety of light sources. 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 $\mathbf{x}$ 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 visibility $V(\mathbf{x}_t)$ for each of these positions can be tested easily with a simple shadow map lookup. While being conceptionally simple, this approach is not very efficient. Various acceleration

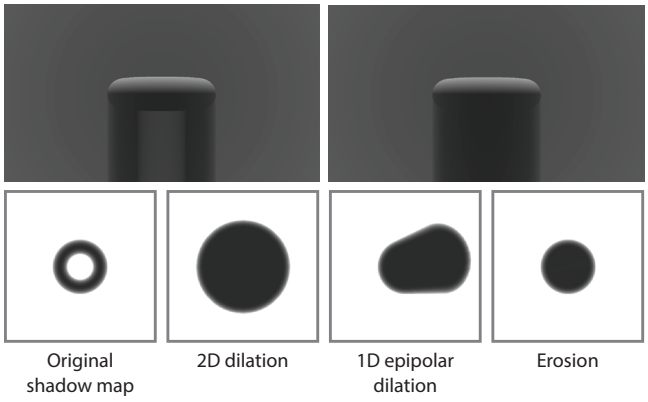

Figure 3: Morphological filtering. Top row: rendering with unmodified and hole-filled shadow map for a simple torus scene with directional light coming straight from the top. Bottom row: shadow maps. From left to right: unmodified input; 2D dilation of input; 1D epipolar dilation of input; successive erosion, which gives equal results for both 1D epipolar and 2D filtering for this particular scene.

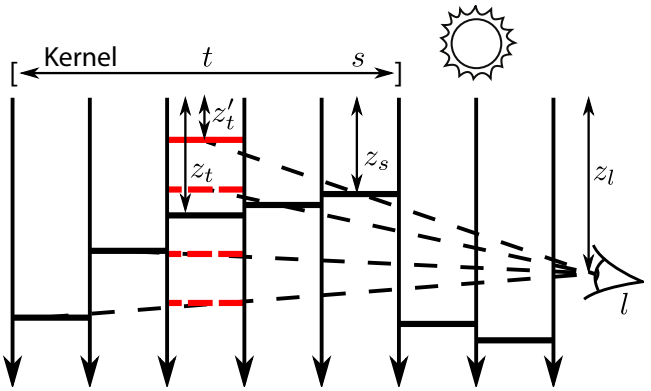

Figure 4: Areas between the parallel light rays represent texels, with the black bars denoting their depth in the shadow map. A 1D kernel (of size 5 in this example) is constructed for texel t with depth z_t. Given the epipole l with depth z_l, the whole kernel is sampled and Eq. 3 is applied. The minimum value z'_t is in this case found for the sample s with depth z_s.

methods [2, 6, 27, 13] have been proposed and we implemented the solution proposed by Chen et al. [6] as well as the method presented by Klehm et al. [13], which work comparably well.

4 Single Scattering Stylization

Our method consists of two major directions to perform scattering stylization, which can also be combined. The first strategy modifies occluders in order to influence their scattering behavior, i.e., light shafts are enhanced, added or removed. The second approach consists in the definition of a transfer function that can be used to drastically influence color, brightness, and contrast. The idea is to rely on values (e.g., average visibility) derived along the view ray to define the final output color. In the following, we will give an overview of these two possibilities.

4.1 Occluder Modification

The main observation is that the appearance of single scattering in a scene largely depends on the number and complexity of light shafts. They become visible due to differences in the average visibility between neighboring view rays (i.e., screen pixels). These differences are often caused by openings in the occluder; that is, holes through which the light can shine, such as openings in a cloud. Our system allows artists to modify the scattering by adding or removing light shafts. The main idea is to edit the shadow map that is used to capture the occluders in the scene. In the following, we will present the various modification options.

Hole Filling The first approach is to simplify the light shafts. This reduces the emphasis of the scattering for an object that may otherwise exhibit a complex scattering appearance, such as a tree. Light shafts are removed by filling holes in the shadow. An extreme example is shown in Fig. 3, where our solution is used to remove the entire hole in the torus, leading to a simpler light shaft appearance.

To fill holes in the occluders, we make use of an image-based approach in which we directly modify the shadow map used for the scattering evaluation. A simple solution for hole filling is the use of 2D morphological filters (see Fig. 3). More precisely, a closure operation can be applied, which consists of a dilation followed by an erosion, both elementary operators. A dilation in the shadow map replaces a value by its minimum in a certain neighborhood, whereas an erosion replaces the value by the maximum. All holes that are removed in this way, also lead to the elimination of the corresponding light shafts. One important factor is the neighborhood to be considered around each pixel, often referred to as the *structural element* or *filter kernel*. Traditionally, a box or circle is used, which we apply as well, additionally giving the user control over the size of the element, which defines the strength of the hole-filling process.

Silhouette Enhancement A very similar approach can be taken to *enhance* light shafts caused by the silhouette of an object. The idea is to extrude objects along the view rays, increasing their thickness. We achieve this by a 1D dilation of the shadow map away from the epipole (see Fig. 3), i.e., the camera position projected into the shadow map. In this way, all new blockers are always hidden by the object itself as the camera only sees the first surface. However, its volumetric shadow *is* increased. The actual enhancement of the light shafts caused by the silhouette of the object is then two-fold. 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 due to sharper boundaries, as illustrated in Fig. 10. Occluders become more drastic and will block more light as they are effectively larger.

The 1D epipolar dilation works as follows. For each texel t we construct a 1D kernel given by the 2D line from t towards the epipole. A standard dilation on the shadow map simply computes the minimum of all samples within the kernel. However, this does not satisfy the required extrusion from the camera position, as the min filter effectively extrudes points in the plane orthogonal to the light direction. It needs to consider the changing z-coordinate along the view ray, which forms a sloped filter kernel. For this, we modify the depth 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 distance between t and epipole l as projected in 2D on the shadow map, and with z_t and z_s the values of t and s in the shadow map, respectively, and z_l the z-coordinate of the epipole l. This boils down to a mix of z_s and z_l modulated by the distance ratio of t to l compared to s to l, which, as per the definition of a dilation, is used only if it is lower than the current lowest depth value z_t. The process is illustrated in Fig. 4

Hole creation Besides removal, we also offer a solution to *add* additional light shafts. In contrast to the previous hole-filling operation, we instead add random holes to the object, the result of which can be seen in Fig. 1. As before, we work directly in image space by

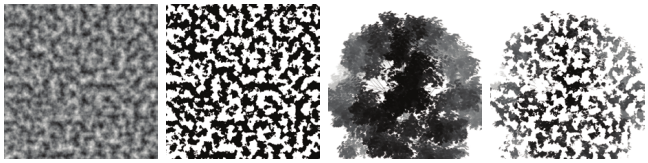

Figure 5: Random hole creation, results of which are shown in Fig. 1. Left to right: Perlin noise with user-defined frequency; thresholding with user-defined hole probability; original shadow map of tree; shadow map with added holes from hole map.

manipulating the shadow map. Each object for which this operation is applied is rendered in a separate shadow map that then undergoes the hole creation process. Basically, a hole is defined by pushing the depth values at its location to one, which corresponds to the far plane. The modified depth maps are then composited with the rest of the scene to yield the depth map for the scattering computation. Alternatively, one could discard fragments of these objects during rasterization, which makes multiple shadow maps unnecessary.

To give control over the hole-creating process, we provide the user with a set of parameters consisting of the average size and the density of holes. In order to avoid perfectly uniform holes, for which the resulting light shafts can look too regular, we make use of Perlin noise [21]. By using a thresholding operation, this scalar noise can be transformed into a binary mask that will be used as the hole map, exhibiting randomization in shape. As Perlin noise is easily parameterizable, users can intuitively steer the hole creation and obtain the desired properties; the number and size of the holes can be influenced via the number of octaves and threshold parameter. The binary mask that is used as the hole map has the value $H(t)$ at the texel with 2D texture coordinates t, and is 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:

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

with o the number of user-defined octaves, p the user-defined persistence and $P(q)$ classical 2D Perlin noise.

The process is illustrated in Fig. 5, where a frequency of $f = 0.3$ and a persistence of 0.5 are used to create 5-octave Perlin noise, which is then thresholded using a hole probability of 0.5, resulting in the hole map shown in the figure.

Using Perlin noise enables easy animation of the hole creation process. To do so, we use a simple two-step lookup. First, a time-based offset is added when sampling the noise. Instead of using the result directly, we re-sample the noise again to prevent the animation from being a simple translation, and effectively randomize the movement. This can simulate the effect of leaves moving in the wind or fake the notion of movement through participating media, both of which we show in our supplemental video.

The hole-creation process is not suitable for all objects. Solid objects for instance can appear unrealistic, and the technique should generally be applied to less recognizable shapes, such as foliage or detailed geometry. Still, it can be applied to solid objects in certain cases to create a sort of halo around a character or simulate the idea of motion, as illustrated in Fig. 2.

4.2 Transfer Functions

An effective way of stylizing scattering is to influence the overall appearance by applying a transfer function (TF) that describes the mapping of properties of a view ray (effectively a pixel), to the scattering component's output color. In order to keep the definition of this transfer function simple, we decided to focus on mapping two parameters to a color. Consequently, the transfer function is defined by a 2D texture, similar to the X-Toon approach [3].

Remembering 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 modifying the TF. As an example, imagine an artist wants a certain set of pixels with similar properties that are currently white, changed to an orange color for stylization purposes. With a simple modification of the transfer function, this becomes easy. Furthermore, using a TF, the general atmosphere due to scattering can be easily transferred from one scene to another.

In practice, we use the average visibility along the view ray and its linearized depth as parameters for influencing the scattering component's color. With those parameters, the result looks physically plausible due to the direct link of the parameters with the actual visibility in the scene. We also experimented with other parameters, such as the average position of visible samples along the view ray, the angle between the view ray and light direction, and the angle between the image x-axis and pixel-to-projected-light direction, but these did not produce meaningful results for the cases we considered, and are therefore not included in our results. Note that we use values along the view ray, i.e., we rely on 3D information, as otherwise, the stylization would appear to be a 2D overlay (shower-door effect), which is directly visible for animated cameras.

To create a TF, we enable artists to interactively design the texture in our framework to explore the various possibilities. This is advantageous over a general image editor as it immediately shows the resulting image, which can be difficult to envision when creating the TF beforehand. The on-the-fly editing of the TF texture uses a painting utility based on layers. In our prototype, solid colors, gradients, images, and diffusion layers are supported, which are optionally blended with each other to produce the final TF texture.

Diffusion layers contain constraints, consisting of a position in the TF texture, a color, and an alpha value. The user can place constraints at any position on the texture, after which they are diffused throughout the layer. One constraint would diffuse to a uniform color for the whole layer, two produce a gradient, while more can create more complex color combinations. For stacked layers, the alpha blending is guided by the constraints' alpha values.

However, specifying constraints only in the 2D parameter space of a TF might not always be intuitive. In order to directly define a desired scattering result at a point in the scene, an artists can simply select a location by clicking in the scene to create a 3D constraint. The parameter space position of that constraint is then computed by projecting its world space position to screen space, querying the pixel's underlying view ray parameters and using these to obtain the new position in the parameter space of the TF.

The selected colors of the constraints are diffused throughout the layer to create a smooth transition in the scene. This results in all pixels in the scene with similar parameters to change accordingly, making it an effective tool for a more global control over the stylization of scattering for similar pixels. 3D constraints are also beneficial for dynamic viewpoints or geometry changes, as we optionally allow the constraint to follow the position of a selected object rather than staying fixed in world space while the camera moves.

The stylization thus far is not expressive enough for modifying the scattering appearance of pre-defined scene animations. We propose a key-framed transfer functions to produce stylized animations. Take for instance an in-game cut-scene with a known camera path. Distinct TF's can be defined for positions along this path, with linear interpolation between them leading to smooth transitions.

Table 1: Performance (in ms) of our prototype for the scene shown in Fig. 1, rendered in full HD for different shadow map (SM) sizes. Measurements for hole creation include Perlin noise creation as well as the SM modification. A transfer function (TF) for color stylization is indexed using the depth map of a GBuffer [23] as well as the view ray average visibility. Using a acceleration method such as the one of Klehm et al. [13] 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	GBuffer	Scattering Vis.	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 used in the scene as shown in Fig. 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 column shows how many holes are filled for the given SM and kernel size.

SM Size	2D Closure		1D 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

Finally, the stylized scattering can be combined with physically-based scattering behavior. For instance, the TF can be monochrome and the stylized scattering modulated by the light source color, which ensures that surface lighting and scattering remain consistent, as in Fig. 7. This enables artists to perform more subtle stylization, like making the scattering more or less dominant. Note that we use the Henyey-Greenstein phase function for the scattering as it makes the result more physically correct and believable. Nonetheless, if desired, the phase function can also be set to a constant.

5 Results and Discussion

Our method works in image space and directly modifies the shadow map that is used for the scattering computation. This has several benefits over working in object space, as the performance does not depend on geometric detail or scene complexity. Furthermore, image-space techniques are efficient on today's hardware and are well-suited for parallel execution. Also, 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 manipulations and pipeline steps for computing and coloring light shafts.

Transfer functions make it possible to achieve very quick and strong changes in the overall appearance of the scattering process, independent of surface, light, or medium parameters. These modifications enable artists to easily redefine the mood of a scene, as illustrated in Fig. 6. Here, the TF creates an alarming atmosphere in the scenes by adding multiple colors with strong edges between them, which causes quantization effects in the resulting scattering. Additionally, while physically-based scattering has an exponential fall-off with respect to the length of the view ray (cf. Eq. 2), the TF used here abandons this fall-off, and more scattering is produced close to the camera. Finally, the figure also illustrates the possibility to pass over a given style from one scene to the next.

Fig. 10 demonstrates the effect of occluder manipulation on the armadillo. Performing a closure operation to fill holes in the shadow map simplifies the scattering appearance. In contrast, an artificial extrusion of the armadillo away from the observer (1D epipolar dilation) enhances edges in the scattering. It causes almost a feeling of fright and emphasizes the armadillo strongly. The resulting effect is similar to the harshness of such effects, often used in comics, to illustrate activeness. As a side effect, the extrusion also darkens the image slightly, due to the larger occluder; a TF could compensate for this (cf. Fig. 7).

The Stanford scene (Fig. 8) again shows our hole-filling procedure, leading to a simplification of the appearance and rather blurry light shaft edges. Note that the 2D filter can cause occlusions to happen in mid-air due to newly created, floating occluders. However, there is little visual impact due to the fact that we integrate along the view ray. While we do not observe artifacts for Fig. 8, they do subtly show in Fig. 10, where in the center image a dark area can be distinguished between the left ear and arm. A solution is to extend a 1D epipolar dilation by a subsequent erosion (Fig. 8, bottom). However, this is not essential for any of our scenes.

Applying the morphological operations to animated scenes requires special care as popping artifacts may be introduced if shadow map holes change in size due to the animation. This can occur when the closure kernel size is not chosen appropriately. If wanted, the kernel size can be key-framed according to the animation.

Hole creation enables artists to easily make the appearance of light shafts more complex. Fig. 9 demonstrates the effect of different parameters on the example scene of Fig. 1. Very small holes (bottom row) can be reduced in size until they average out due to the integration along the view ray, while very large holes (top row) can drastically simplify the scattering behavior. The control parameters make it easy to transition between more physically-based scattering with the original occluders, and the more exaggerated alternatives. Note that the axes are not absolute; the number of large holes is less than for smaller holes in the same column. The scene can also be animated, e.g., imagine a sunrise for Fig. 1. The inherent complexity of the tree masks any potential artifacts that one may expect from the 2D hole map, which is fixed to the light space. In general, we expect the hole creation to be applied to objects of similar complexity, for which the shadows are not easy to predict.

Finally, the turtle scene (Fig. 7) shows an example for a combination of occluder manipulation and stylization via a TF. The initial shot exhibits an unlucky overlap of occluders (head, body, fins). An artist may want to simplify the scattering to put more emphasis on the actual object in the scene and remove attention from the light shafts. The 2D morphological filtering closes the hole between the fins of the turtle and smooths the occluder's silhouette, which gives the light shafts a more simplified appearance. Then, the applied TF (Fig. 7, right) makes it possible to reduce the increased shadows, while keeping a plausible scattering appearance; the TF approximates physically-based scattering, having a gradient in both axes.

Additional examples and also animated results are shown in the supplemental video.

6 Conclusion

We have presented strategies that let artists stylize volumetric single scattering in various expressive ways. The stylization can be easily transferred to different scenes and also works for animations.

Our image-based occluder manipulations modify the complexity of the scattering appearance. Light shafts are added using hole creation, which is randomized but controlled with a few parameters. Simplification of light shafts and silhouette enhancement are achieved by morphological filter operations.

We also propose the usage of transfer functions to control the final appearance and to transport the general mood of scattering ef-

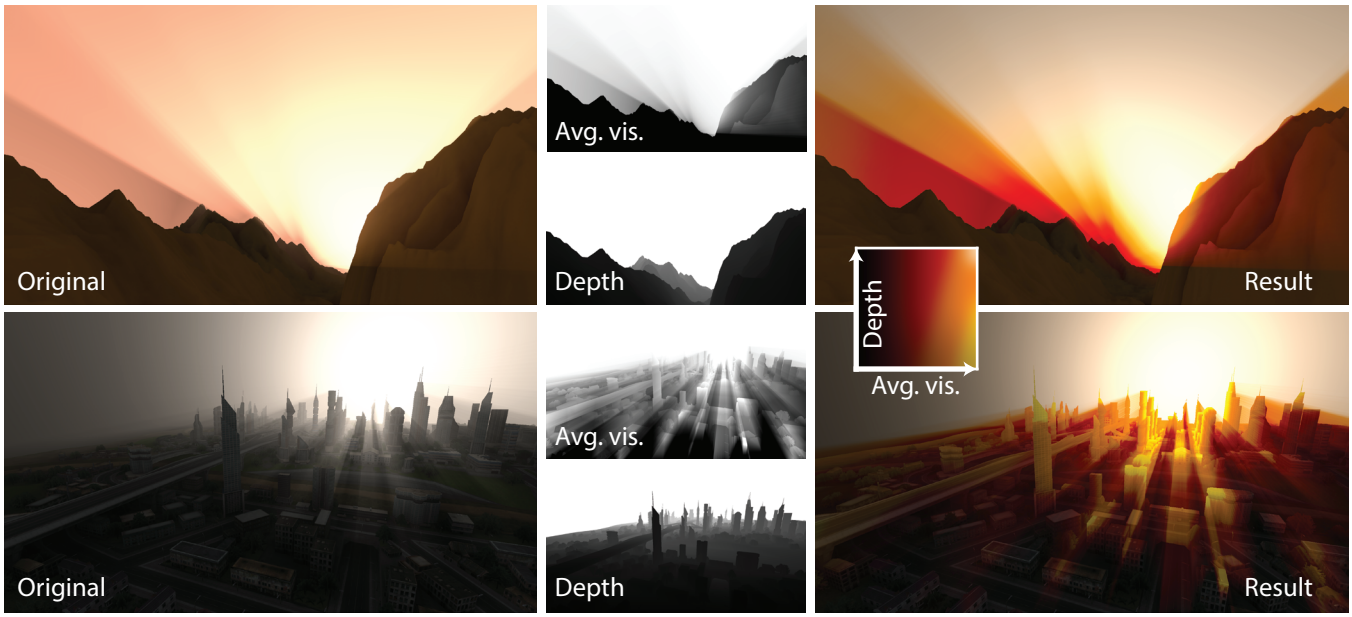

Figure 6: Expressiveness of transfer functions. Physically-based scattering (left) is stylized (right) using a transfer function (inset), which is parametrized by the average visibility and linearized depth of the view rays (center).

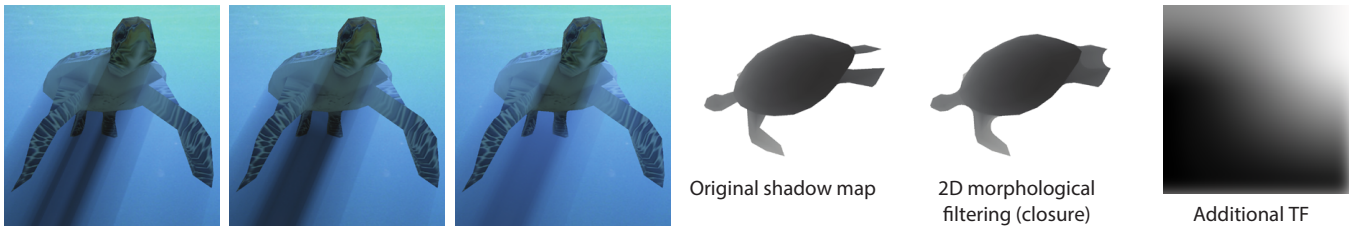

Figure 7: Combined use of occluder manipulation and a TF. From left to right: the original image; hole filling unifies the fins of the turtle; a simple TF is applied to reduce the darkening for long view rays; original shadow map; hole-filled shadow map; applied TF.

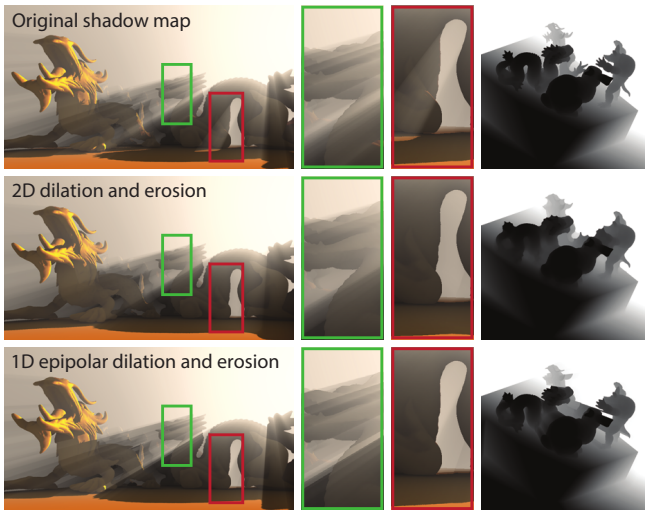

Figure 8: Object simplification by hole filling. From top to bottom: scattering with original shadow map; modified shadow map by 2D morphological filtering (a closure operation); modified shadow map by a 1D epipolar closure.

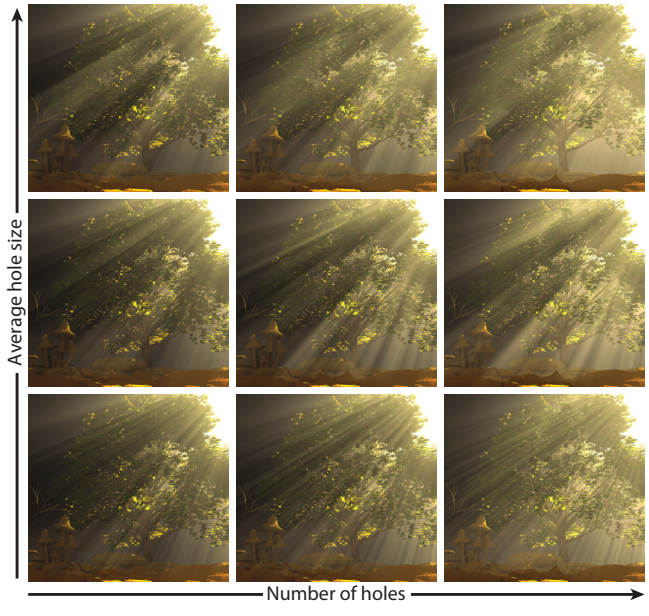

Figure 9: Effect of noise parameters on the resulting scattering.

Figure 10: Object simplification by hole filling. From left to right: scattering with original shadow map; modified shadow map by 2D morphological filtering consisting of a dilation and erosion, i.e., a closure operation; modified shadow map by object extrusion through 1D epipolar dilation.

fects. By parameterizing the transfer function with the average visibility and linear depth, results look physically plausible, yet expressiveness remains. Transfer functions are modified by artists interactively, where we enable them to stylize the scattering for similar view rays and to animate the atmosphere of a scene over time.

Our solution comes at a low computational cost, which makes it ready to use for real-time applications, enabling a quick exploration of the various settings. Our work makes a first step in the direction of a general scattering stylization, and its ability to reproduce various scenarios that would otherwise require laborious manual tweaking, makes it an interesting addition to the artistic toolbox.

Interesting future work could include additional object-focus strategies, heterogeneous media, multiple scattering, and advanced TF editing, e.g., inferring TF parameters from user-drawn strokes.

Acknowledgements

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

References

[1] M. Ament, F. Sadlo, C. Dachsbacher, and D. Weiskopf. Low-pass filtered volumetric shadows. *IEEE TVCG*, 20(12):2437–2446, 2014.

[2] I. Baran, J. Chen, J. Ragan-Kelley, F. Durand, and J. Lehtinen. A hierarchical volumetric shadow algorithm for single scattering. *ACM TOG (SIGGRAPH Asia)*, 29(6):178:1–178:10, 2010.

[3] P. Barla, J. Thollot, and L. Markosian. X-toon: An extended toon shader. In *NPAR*, pages 127–132, 2006.

[4] M. Billeter, E. Sintorn, and U. Assarsson. Real time volumetric shadows using polygonal light volumes. In *HPG*, pages 39–45, 2010.

[5] S. Calderon and T. Boubekeur. Point morphology. *ACM TOG (SIGGRAPH)*, 33(4):45:1–45:13, July 2014.

[6] J. Chen, I. Baran, F. Durand, and W. Jarosz. Real-time volumetric shadows using 1d min-max mipmaps. In *i3D*, pages 39–46, 2011.

[7] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella. Directing gaze in 3d models with stylized focus. In *EGSR*, pages 377–387, 2006.

[8] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz. Stylized shadows. In *NPAR*, pages 77–83, 2007.

[9] O. Elek, T. Ritschel, C. Dachsbacher, and H.-P. Seidel. Interactive light scattering with principal-ordinate propagation. In *GI*, 2014.

[10] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE TVCG*, 17(12):2106–2114, 2011.

[11] W. B. Kerr, F. Pellacini, and J. D. Denning. Bendylights: Artistic control of direct illumination by curving light rays. *Comp. Graph. Forum (EGSR)*, 29(4):1451–1459, 2010.

[12] O. Klehm, I. Ihrke, H.-P. Seidel, and E. Eisemann. Volume stylizer: Tomography-based volume painting. In *i3D*, pages 161–168, 2013.

[13] O. Klehm, H.-P. Seidel, and E. Eisemann. Prefiltered single scattering. In *i3D*, pages 71–78, 2014.

[14] S. Lee, E. Eisemann, and H.-P. Seidel. Depth-of-field rendering with multiview synthesis. *ACM TOG (SIGGRAPH Asia)*, 28(5):134:1–134:6, 2009.

[15] O. Mattausch, T. Igarashi, and M. Wimmer. Freeform shadow boundary editing. *Comp. Graph. Forum (EG)*, 32:175–184, 2013.

[16] D. Nowrouzezahrai, J. Johnson, A. Selle, D. Lacewell, M. Kaschalk, and W. Jarosz. A programmable system for artistic volumetric lighting. *ACM TOG (SIGGRAPH)*, 30(4):29:1–29:8, 2011.

[17] 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)*, 27(4):1217–1223, 2008.

[18] D. Patel, V. Šoltészová, J. M. Nordbotten, and S. Bruckner. Instant convolution shadows for volumetric detail mapping. *ACM TOG*, 32(5):154:1–154:18, 2013.

[19] V. Pegoraro and S. G. Parker. An analytical solution to single scattering in homogeneous participating media. *Comp. Graph. Forum (EG)*, 28(2):329–335, 2009.

[20] 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)*, 24(3):464–470, 2005.

[21] K. Perlin. An image synthesizer. *Computer Graphics (SIGGRAPH)*, 19(3):287–296, 1985.

[22] 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)*, 26(3):25:1–25:11, 2007.

[23] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. *Computer Graphics (SIGGRAPH)*, 24(4):197–206, Sept. 1990.

[24] J. Schmid, R. W. Sumner, H. Bowles, and M. Gross. Programmable motion effects. *ACM TOG (SIGGRAPH)*, 29(4):57:1–57:9, 2010.

[25] 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)*, 32(4):129, 2013.

[26] Y. Song, X. Tong, F. Pellacini, and P. Peers. Subedit: A representation for editing measured heterogeneous subsurface scattering. *ACM TOG (SIGGRAPH)*, 28(3):31, 2009.

[27] C. Wyman. Voxelized shadow volumes. In *HPG*, pages 33–40, 2011.