Property and Lighting Manipulations for Static Volume Stylization Using a Painting Metaphor

Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, Elmar Eisemann

Property and Lighting Manipulations for Static Volume Stylization Using a Painting Metaphor

Oliver Klehm Ivo Ihrke Hans-Peter Seidel Elmar Eisemann

Abstract—Although volumetric phenomena are important for realistic rendering and can even be a crucial component in the image, the artistic control of the volume's appearance is challenging. Appropriate tools to edit volume properties are missing, which can make it necessary to use simulation results directly. Alternatively, high-level modifications that are rarely intuitive, e.g., the tweaking of noise function parameters, can be utilized.
Our work introduces a solution to stylize single-scattering volumetric effects in static volumes. Hereby, an artistic and intuitive control of emission, scattering and extinction becomes possible, while ensuring a smooth and coherent appearance when changing the viewpoint. Our method is based on tomographic reconstruction, which we link to the volumetric rendering equation. It analyzes a number of target views provided by the artist and adapts the volume properties to match the appearance for the given perspectives. Additionally, we describe how we can optimize for the environmental lighting to match a desired scene appearance, while keeping volume properties constant. Finally, both techniques can be combined. We demonstrate several use cases of our approach and illustrate its effectiveness.

Index Terms—artist control, optimization, participating media

1 Introduction

Computer graphics allows us to depict virtual worlds with stunning visual complexity by simulating laws of nature in combination with an accurate description of a virtual scene. Volumetric phenomena [1], [2], [3], [4] are an important element to make synthetic scenes appear richer and less sterile. Usually, participating media are often represented by voxels, which store physical properties such as extinction, absorption, and scattering behavior. Nonetheless, it is difficult to select the right parameters for each volume element. Often, volumes are populated by means of simulation [5] or procedural models [6], but such solutions remain non-intuitive and do not allow for simple fine-grained appearance control. Although specialized approaches have been introduced for shape control [7], [8], an intuitive control of the volume's parameters to ensure a certain appearance under complex illumination conditions is missing.

It is true that physical accuracy is not necessarily mandatory for a convincing image, but it is difficult to achieve plausible results without a proper physical basis. For this reason, artists often start with a realistic simulation before modifying the appearance, e.g., [9], [10], [11]. Unfortunately, changing the appearance of volumes is difficult. Due to transparency, changes applied in one view usually affect the entire data set in a non-intuitive manner. Hence, matching a certain appearance while maintaining a realistic overall look is difficult.

We investigate appearance control and let the user modify a static volume directly to match its appearance for certain viewpoints using familiar image editing operations. The user can be completely oblivious of physical models and does not need to estimate the influence of environmental illumination. All that needs to be provided is a drawing of the desired scene appearance for some viewpoints and our algorithm optimizes the physically-based properties (albedo, emission, or -under special conditions- extinction) in order to match the appearance as well as possible. Alternatively, the user can also let the system optimize the environmental illumination, or both. For the adjustment of the volume's properties, our solution is limited to single-scattering. Nevertheless, even with this restriction many important cues are captured and images remain convincing. This choice also ensures that interactive editing sessions become possible with update times in the order of seconds.

The contributions of this paper are as follows:

- Based on the radiative transport equation, we derive conditions under which static volume-appearance stylization via a fast linear optimization is possible (Sec. 4.1).
- The implementation of our approach is efficient in terms of execution time and memory cost (Sec. 5).
- We present an environmental lighting optimization that can be coupled with the volume stylization (Sec. 4.4).

- *O. Klehm and H.-P. Seidel are with MPI Informatik, 66123 Saarbrücken, Germany. E-mail: {oklehm, hps}@mpi-inf.mpg.de.*
- *I. Ihrke is with Inria Bordeaux Sud-Ouest, France. E-mail: ivo.ihrke@inria.fr.*
- *E. Eisemann is with Delft University of Technology, Netherlands. E-mail: E.Eisemann@tudelft.nl.*

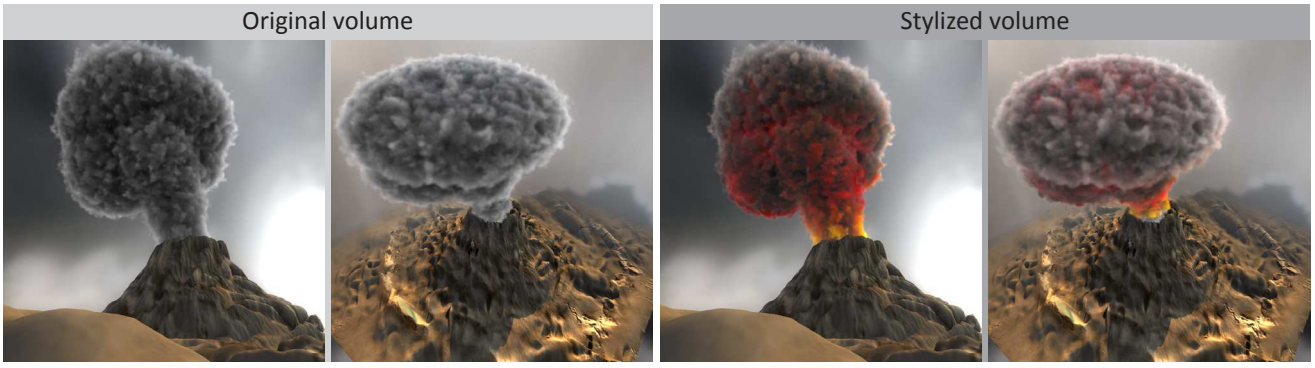

Fig. 1. Volume stylization of an environmentally lit, static smoke of an volcano. On the left, we show the original volume model, whereas on the right, the volume was stylized to increase the atmospheric tension of the scene. The modifications include a yellow to red to gray color gradient, and a red color cast, that mimics an active glow from the inside. We further added irregular red/yellow stripes that are commonly used in comics to give the impression of ongoing dynamics. Our technique optimizes for the static volume properties emission and albedo from a handful of user defined images. Rendering the new volume reveals details following the user input such as the glow from the center of the volume.

In this paper, we extend existing work on volume stylization, published at the ACM Symposium on Interactive 3D Graphics and Games [12]. With respect to this previous publication, we present several novelties in this article. We show how our solution can be used to optimize the environmental lighting while keeping the volume properties constant (Sec. 4.4). Previously, such optimizations have only been applied in the context of surface-based representations [13]. Here, we show how to extend these optimizations to the context of volume data. We also analyze the applicability of the stylization method to animated scenes.

2 Related Work

Producing realistic volumetric data sets is known to be difficult, which is one of the reasons for the existence of various methods that capture properties of natural phenomena; flames [14], smoke [15], [16], or refractive elements [17], [18]. Often, the process involves tomographic reconstruction [14], [16], [18]. It is surprising, that even a small number of views (8 to 16) often leads to a convincing volumetric description. This is one of our motivations to define volume properties in an image-based fashion. Nonetheless, previous work employed simple image formation models and restricted the capturing process to a single phenomenon such as emission [14], [16], or refraction [18]. One exception is the work by Lintu et al. [19]. They attempt to recover emission and absorption simultaneously to reconstruct a nebula from astronomical observations. In contrast, one of our goals is to artistically control and modify volumes to achieve a certain appearance under complex lighting.

Even for a skilled artist, it is often very difficult to tweak the appearance of a scene without destroying its plausible appearance. For this reason, many artistic modifications start from a physical simulation that is modified until the desired appearance is achieved. Unfortunately, these modifications are often complicated because physically-based parameters can be non-intuitive and the physical models complex. More intuitive manipulation means have been investigated for light-source editing [20] or modifications to the light transport of spot lights [10], indirect light [9], and shadows [21].

Participating media have received little attention due to the very complex relationships between the volume's properties and its final rendering. An approach for sub-surface scattering [22] exists and the results can look convincing, yet the scattering needs to be relatively strong and objects should be rather opaque. Light beams [11] also include volumetric effects, but the focus is on light modification, not on changing the properties of the medium. The approach allows even non-physical modifications, such as curved rays. Nonetheless, the volume-rendering process is split into multiple functions that are changed individually instead of a global optimization procedure. Finally, both methods also rely on specialized rendering routines, whereas we target the modification of the properties of a volume. Hereby, the influence on appearance is consistent with the radiative transport equation [23]. Hence, standard rendering techniques can be employed with our solution. Recently, Hašan and Ramamoorthi [24] presented an approximation to quickly re-render an image under full light transport after a change of the albedo of a volume. While they share some goals with our work, they do not present a general approach to edit the actual albedo parameters of a volume. Further, the approximation is limited to dense volumes and requires a costly and memory intensive preprocess.

In the field of scientific visualization, transfer functions have been used and studied to get a better un-

derstanding of the underlying data and reveal hidden information. While methods in scientific visualization share a similar goal of appearance change [25], [26], they iteratively update the transfer function based on direct user interactions. In contrast, our method builds on an inverse approach to derive physical volume parameters that match target images when rendered.

More closely related is the problem of style-transfer from a photograph to a 3D scene. Some approaches rely on heuristics and an exhaustive search to discover good material properties [27] or are limited to a certain type of scene [28]. The latter work is similar in its goals to our work, in the sense that the parameters of an artificial cloud are to be modified. In their work, a style is transferred from a photograph to a single view-dependent rendering. In contrast to our solution, the authors aim at estimating a few global volume parameters in a non-linear optimization scheme instead of estimating per-voxel properties.

One could also consider the modification of the environmental illumination to optimize the volume's appearance instead of the volume itself. While previous work showed such inverse rendering solutions for surface-based scenes [13], [29], we propose an efficient solution for volume data. A recent example for inverse rendering derives light-source parameters based on user input for a fixed number of light sources [30]. Again, the optimization is non-linear and treated as a black box that uses rendering as an evaluation process to uncover the optimal parameter set. In general, it is important to realize that non-linear functions are often difficult to optimize. They often suffer from local minima and, due to numerical differentiation, certain limits exist in the number of parameters that can be optimized for. Usually, only a slow convergence is achieved. In our solution, we opt for a linear optimization to ensure a fast execution of our algorithm.

Our work also shares some characteristics with recent developments in fabrication. Previous work showed solutions to built physical shapes that have a predefined way of interacting with light in the real world. These methods make it possible to produce objects that cast certain shadows [31], [32], caustics [33], [34], or exhibit a certain surface reflectance [35]. Such work also relates to light-field display technology [36], [37], 6D displays [38], or other special displays [39], [40] that make it possible to achieve a specific appearance when uniformly illuminating from the back and observing under particular viewing angles.

In contrast to our work, these methods consider only a small number of three to five layered light modulating planes which are viewed from a well-defined viewing zone. Further, the approaches involve only one type of effect, such as absorption [39], [40] or change of polarization [36]. Our goal is to give control over appearance of volumes having a full resolution along all three spatial dimensions. We consider complex and full surround views and complex environmental illumination. Further, we consider many properties of the volume, which can be easily defined for our rendering context, but that would lead to problems for physical object manufacturing. One particular example are refracted rays that we can trace through the known volume.

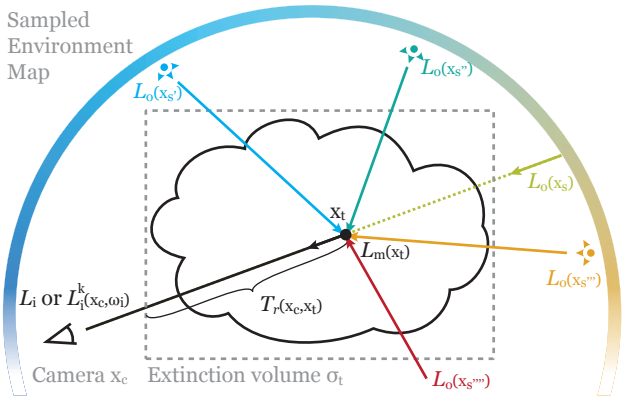

Fig. 2. Single scattering: Radiance is accumulated along the view ray. At each sample point $\mathbf{x}_t$ the emitted and incoming attenuated light is scattered towards the observer ($L_\mathrm{m} = L_\mathrm{emit} + L_\mathrm{scat}$) positioned at $\mathbf{x}_c$ along a ray with direction $-\omega_i$.

3 Background

This section covers the necessary background of our approach; we review the volumetric rendering equation and show that it can be inverted under certain conditions. The inversion leads to a tomographic problem which is solved by inverting a large-scale linear system.

We start by reviewing volumetric rendering. This process simulates how light interacts with participating media. Further, we will introduce the main volume properties that have an effect on the appearance. This background will be useful to understand how we offer control over appearance.

Rendering complex light transport in participating media is done by solving the radiative transport equation [23]. It applies the physical scattering cross section model, a hypothetical area that describes the likelihood of light being absorbed, scattered, or emitted in a specific region of the volume. For the moment, we do not consider refraction, which will be discussed in Sec. 5.2. We summarize all symbols in Tab. 1.

The emission $L_\mathrm{emit}(\mathbf{x})$, albedo $\rho(\mathbf{x})$, and extinction $\sigma_\mathrm{t}(\mathbf{x})$ coefficients of a volume V describe its optical properties with sufficient accuracy for high quality volume rendering. In the following, we therefore formulate volume rendering in the context of these three spatially-dependent volume parameters that we will later optimize for (Sec. 4):

The light incident at a point in the scene $\mathbf{x}$ (e.g., the camera position) from direction ω_i consists of two

Symbol	Volume properties
σ_t	Extinction coefficient (in range $[0;\infty]$, it holds: $\sigma_\mathrm{t} = \sigma_\mathrm{s} + \sigma_\mathrm{a}$)
σ_s	Scattering coefficient (likelihood to scatter incoming light)
σ_a	Absorption coefficient (likelihood to absorb incoming light)
ρ	Albedo, i. e., $\sigma_\mathrm{s}(\mathbf{x}) = \sigma_\mathrm{t}(\mathbf{x})\,\rho(\mathbf{x})$.
L_emit	Radiance emitted by the volume
f	Scattering phase function
	Description
$\mathbf{x}, \mathbf{x}_\mathrm{s}$	Position, background surface hit by view ray or infinity
$\omega, \omega_{\mathrm{i/o}}, \Omega$	Direction, incoming/outgoing, unit sphere of directions
$L_{\mathrm{i/o}}$	Radiance incoming/outgoing
L_m	Medium radiance outgoing from volume point
$T_r(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{b})$	$:= e^{-\int_c \sigma_\mathrm{t}(\mathbf{x}_t)dt}$ - Transmittance or probability, that a photon travels along an arc-length parameterized curve c from $\mathbf{x}_\mathrm{a}$ to $\mathbf{x}_\mathrm{b}$

TABLE 1
Symbols used for volume rendering

terms:

$$L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i}) = T_r(\mathbf{x}, \mathbf{x}_\mathrm{s})\, L_\mathrm{o}(\mathbf{x}_\mathrm{s}, -\omega_\mathrm{i}) + \int_c T_r(\mathbf{x}, \mathbf{x}_t)\, \sigma_\mathrm{t}(\mathbf{x})\, L_\mathrm{m}(\mathbf{x}_t, -\omega_\mathrm{i})\, dt. \quad (1)$$

The first summand is the reflected radiance at the first visible surface or the emitted radiance from the background attenuated by the out-scattering due to the volume, $T_r(\mathbf{x}, \mathbf{x}_\mathrm{s})$. The second is the medium radiance $L_\mathrm{m}(\mathbf{x}, \omega_\mathrm{o})$, which is emitted directly or in-scattered at each point $\mathbf{x}_t$ along the light path c and again attenuated by the volume. Consequently, we split $L_\mathrm{m}(\mathbf{x}, \omega_\mathrm{o})$ into emission and in-scattering:

$$L_\mathrm{m}(\mathbf{x}, \omega_\mathrm{o}) = L_\mathrm{emit}(\mathbf{x}, \omega_\mathrm{o}) + \rho(\mathbf{x})\, L_\mathrm{scat}(\mathbf{x}, \omega_\mathrm{o}). \quad (2)$$

The outgoing direction at the points $\mathbf{x}_s$ and $\mathbf{x}_t$ is defined as the direction towards the scene point $\mathbf{x}$, i.e., $\omega_o = -\omega_i$. Emission is the simpler part; the likelihood of a volume element to emit a fixed radiance is independent of direction and regardless of other properties of the volume:

$$L_\mathrm{emit}(\mathbf{x}, \omega_\mathrm{o}) = L_\mathrm{emit}(\mathbf{x}). \quad (3)$$

The albedo is multiplying the in-scattering integral, properly modulating the scattered light by the likelihood that scattering and not absorption occurs. Determining in-scattering requires integrating over the entire unit sphere surrounding the point to gather incoming light, which is then modulated by the phase function (the volumetric equivalent to BRDFs for surfaces; while not limited by our method, in practice, we use an isotropic phase function for convenience).

$$L_\mathrm{scat}(\mathbf{x}, \omega_\mathrm{o}) = \int_\Omega f(\mathbf{x}, (\omega_\mathrm{i} \cdot \omega_\mathrm{o}))\, L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i})\, d\omega_\mathrm{i}.$$

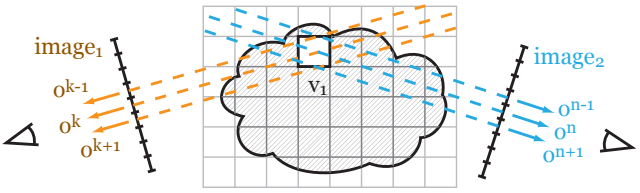

Fig. 3. Volume from two viewpoints. Problem: Changing one voxel v_1 influences several pixels in $image_1$ ($\mathrm{o}^{\mathrm{k}-1}, \mathrm{o}^{\mathrm{k}}, \mathrm{o}^{\mathrm{k}+1}$) and in $image_2$ ($\mathrm{o}^{\mathrm{n}-1}, \mathrm{o}^{\mathrm{n}}, \mathrm{o}^{\mathrm{n}+1}$)

To compute the incoming light $L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i})$ in $L_\mathrm{scat}(\mathbf{x}, \omega_\mathrm{o})$, one has to apply Eq. 1 recursively, which makes volume rendering hard. For simplification, multiple volume scattering is often ignored, i. e., $L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i}) := T_r(\mathbf{x}, \mathbf{x}_\mathrm{s})\, L_\mathrm{o}(\mathbf{x}_\mathrm{s}, \omega_\mathrm{i})$; only light originating outside the volume (environmental, direct light sources…) is considered, see Fig. 2. In the following, we also apply this approximation.

4 Volume & Light Reconstruction

To control appearance, we want the user to define a single or multiple images describing the desired target views of the volume. For these images, we invert the volume-rendering process to optimize for the properties, describing the volume. We treat images as a collection of N *constraint pixels*. Given the corresponding camera view, a pixel with index $k \in [1; N]$ corresponds to a ray (origin $\mathbf{x}^k$ and direction ω^k). We denote its value $L_\mathrm{i}^k(\mathbf{x}^k, \omega^k)$ and, if applicable, the position of the first hit surface $\mathbf{x}_\mathrm{s}^k$, which otherwise is the background at ∞. We want to modify properties of a volume V, such that rendering with V, yielding $L_\mathrm{i}(\mathbf{x}^k, \omega^k)$, matches the *pixel constraint*, $L_\mathrm{i}(\mathbf{x}^k, \omega^k) = L_\mathrm{i}^k(\mathbf{x}^k, \omega^k)$. For example, one could modify the emission field of V to match a given appearance.

4.1 Volume Reconstruction

A single pixel constraint can influence many voxels and, inversely, two pixel constraints might imply changes on one and the same voxel. We illustrate this situation in Fig. 3. Consequently, a perfect solution might not always be possible.

Instead, we seek to find a coefficient vector $\mathbf{a} := (\ldots, a^i, \ldots)$ such that a linear combination $\sum_i a^i\, v^i(\mathbf{x})$ of known basis functions v^i defines a property of the volume such that the constraint pixels are matched best.

From a mathematical point of view, the basis functions could have global support. However, in practice, using a basis with spatially local support speeds up the reconstruction. One convenient choice for a basis are box functions associated to the volume's voxels, which corresponds to nearest-neighbor sampling of a 3D texture. Using triangle functions allows us to consider linearly interpolated solutions as well.

Next, we will show that the best match for emission, scattering, and extinction can be obtained by solving a linear system

$$\mathbf{o} = \mathbf{W}\mathbf{a}, \tag{4}$$

where $\mathbf{a}$ are the basis coefficients to be computed. The observations $\mathbf{o} := (\dots, o^k, \dots)^{\mathrm{T}}$ involve the constraint pixel values, and the matrix $\mathbf{W} := (\dots, \mathbf{w}^{ik}, \dots)^{\mathrm{T}}$ is derived from the volume rendering equation.

4.2 Property Reconstruction

The volume parameter reconstruction implies that Eq. 1 needs to be linearized. In the following, we derive the entries of matrix $\mathbf{W}$ that enable the estimation of specific volume properties. The derivation is carried out for a single constraint pixel, i. e. , for one row of matrix $\mathbf{W}$.

We isolate volume properties that can be linearly optimized. All other volume properties are assumed to be fixed and summarized into the coefficients of matrix $\mathbf{W}$ and the observation vector $\mathbf{o}$, respectively.

4.2.1 Emission

The derivation starts with Eq. 1 in conjunction with Eqs. 2 and 3. For a constraint pixel k, we obtain:

$$L_{\mathrm{i}}^k(\mathbf{x}^k, \omega^k) = T_r(\mathbf{x}^k, \mathbf{x}_{\mathrm{s}}^k)\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k) + \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, (L_{\mathrm{emit}}(\mathbf{x}_t) + \rho(\mathbf{x}) L_{\mathrm{scat}}(\mathbf{x}_t, \omega^k))\, dt.$$

The integral is carried out along a straight ray c connecting the view position $\mathbf{x}^k$ and the first point on the background $\mathbf{x}_s$. We will later describe how refracted rays can be incorporated (Sec. 5.2).

Assuming single scattering, we can split the integral into emitted and scattered light and unify all fixed values in o^k:

$$\begin{aligned} o^k &:= L_{\mathrm{i}}^k(\mathbf{x}^k, \omega^k) - T_r(\mathbf{x}^k, \mathbf{x}_{\mathrm{s}}^k)\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k) - \\ &\qquad \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, \rho(\mathbf{x})\, L_{\mathrm{scat}}(\mathbf{x}_t, \omega^k)\, dt \\ &= \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, L_{\mathrm{emit}}(\mathbf{x}_t)\, dt. \end{aligned} \tag{5}$$

We represent $L_{\mathrm{emit}}(\mathbf{x}_t)$ by a linear combination of basis functions, whose integral can be computed:

$$\begin{aligned} o^k &= \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t) \Big(\sum_i L_{\mathrm{emit}}^i\, \upsilon^i(\mathbf{x}_t) \Big)\, dt \\ &= \sum_i L_{\mathrm{emit}}^i \Big(\int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, \upsilon^i(\mathbf{x}_t)\, dt \Big) \\ &=: (\vec{L}_{\mathrm{emit}} \cdot \mathbf{w}^k). \end{aligned} \tag{6}$$

The coefficients of the above equation are defined by an integral corresponding to *single* ray passing through the volume. Combining all equations defined by the constraint pixels, we obtain a linear system. The coefficient vector is $\mathbf{a} = \vec{L}_{\mathrm{emit}} = [L_{\mathrm{emit}}^1 \ \dots \ L_{\mathrm{emit}}^M]^{\mathrm{T}}$ in this case with M as the number of unknowns and basis functions.

4.2.2 Albedo

For albedo optimization, we can establish a similar linearization as for emission. Combining volume rendering (Eq. 1) and the constraints from a pixel k, we obtain - similar to Eq. 5:

$$\begin{aligned} o^k &:= L_{\mathrm{i}}^k(\mathbf{x}^k, \omega^k) - T_r(\mathbf{x}^k, \mathbf{x}_{\mathrm{s}}^k)\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k) - \\ &\qquad \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, L_{\mathrm{emit}}(\mathbf{x}_t)\, dt \\ &= \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, L_{\mathrm{scat}}(\mathbf{x}_t, \omega^k)\, \rho(\mathbf{x}_t)\, dt. \end{aligned}$$

With single scattering, $L_{\mathrm{scat}}(\mathbf{x}_t, \omega^k)$ is independent of ρ, and we can solve for the latter. More precisely, representing the field of ρ with a linear combination of basis functions, we obtain:

$$\begin{aligned} o^k &= \sum_i \rho^i \Big(\int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, L_{\mathrm{scat}}(\mathbf{x}_t, \omega^k)\, \upsilon^i(\mathbf{x}_t)\, dt \Big) \\ &=: (\vec{\rho} \cdot \mathbf{w}^k). \end{aligned}$$

Again, the coefficients are defined via an integral that translates to a ray marching process involving the known properties of the volume and the coefficient vector is $\mathbf{a} = \vec{\rho}$ in this case.

4.2.3 Emission & Albedo

The combination of emission and albedo can be jointly optimized because L_{emit} and ρ are linearly independent, i.e., entering as two different summands in Eq. 2. In this case, the coefficient vector becomes $\mathbf{a} = [\vec{L}_{\mathrm{emit}}; \vec{\rho}]$.

4.2.4 Extinction

The extinction coefficient can only be reconstructed if we assume that the volume's outgoing radiance is constant, i. e. , $L_{\mathrm{m}}(\mathbf{x}, \omega) = L_{\mathrm{m}}(\mathbf{x}', \omega') = const.\ \forall \mathbf{x}', \omega'$. In this case, our problem becomes similar to computed tomography (compare Sec. 4.3). As extinction is mostly used to define the overall shape of the volume and usually the first property to be derived, this restriction is usually not too problematic (Sec. 4.3). Starting with Eq. 1 and expanding T_r:

$$L_{\mathrm{i}}^k(\mathbf{x}^k, \omega^k) = e^{-\int_c \sigma_{\mathrm{t}}(\mathbf{x}_t) dt} L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k) + \int_c e^{-\int_c \sigma_{\mathrm{t}}(\mathbf{x}_t) dt}\, \sigma_{\mathrm{t}}(\mathbf{x}_{t'})\, L_{\mathrm{m}}(\mathbf{x}_{t'}, \omega^k)\, dt'.$$

We employ the linear combination of basis functions and can rewrite the first summand:

$$L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k)\, e^{-\int_c \sum \sigma_{\mathrm{t}}^i \upsilon^i(\mathbf{x}_t) dt} = L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \omega^k) \prod_i e^{-\sigma_{\mathrm{t}}^i \int_c \upsilon^i(\mathbf{x}_t) dt}. \tag{7}$$

Concerning the second summand, we exploit the constant outgoing radiance and remove L_{m} from the integral. The remainder can be integrated, yielding $1 - e^{-\int_0^s \sigma_{\mathrm{t}}(\mathbf{x}_t) dt}$. Mathematically, the result can be shown by decomposing σ_{t} into a piecewise-constant

approximation and splitting the outer integral accordingly [2]. Then each integral can be solved and the result recombined. This proof is valid for any Riemann-integrable extinction function. The proof also follows logically; the above remainder is the probability that a ray from the camera passes through the volume without hitting a particle, so it is one minus the probability that a ray is stopped. Now, we can use a transformation similar to the first summand to obtain:

$$\begin{aligned} L_{\mathrm{i}}^k(\mathbf{x}^k,\omega^k) = \quad & L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k,\omega^k)\prod_i e^{-\sigma_{\mathrm{t}}^i\int_c \upsilon^i(\mathbf{x}_t)dt} + \\ & L_{\mathrm{m}}\left(1-\prod_i e^{-\sigma_{\mathrm{t}}^i\int_c \upsilon^i(\mathbf{x}_t)dt}\right) \\ = L_{\mathrm{m}} + (L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k,\omega^k) - L_{\mathrm{m}})&\prod_i e^{-\sigma_{\mathrm{t}}^i\int_c \upsilon^i(\mathbf{x}_t)dt}. \end{aligned}$$

Finally, we apply a logarithm to obtain a linear system in σ_{t}^i:

$$\begin{aligned} o^k := \ln\Big(\frac{L_{\mathrm{i}}^k(\mathbf{x}^k,\omega^k)-L_{\mathrm{m}}}{L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k,\omega^k)-L_{\mathrm{m}}}\Big) &= \ln\Big(\prod_i e^{-\sigma_{\mathrm{t}}^i\int_c \upsilon^i(\mathbf{x}_t)dt}\Big) \\ = \sum_i \ln(e^{-\sigma_{\mathrm{t}}^i\int_c \upsilon^i(\mathbf{x}_t)dt}) &= \sum_i -\sigma_{\mathrm{t}}^i\Big(\int_c \upsilon^i(\mathbf{x}_t)dt\Big). \end{aligned}$$

Gathering all extinction coefficients σ_{t}^i in a vector $\vec{\sigma_{\mathrm{t}}}$, the coefficient vector of the linear system becomes $\mathbf{a} = \vec{\sigma_{\mathrm{t}}}$.

4.3 Discussion of Volume Reconstruction

In all three cases that we discussed previously, we start with the volume rendering integral that contains the volume property of interest. The target field is represented as a linear combination of basis functions, which allows us to move the coefficients out of the integral, and, hereby, to isolate the unknowns. The same process is applied in computed tomography [41], which corresponds to our reconstruction of the extinction coefficient. However, different from medical computed tomography, we have to deal with *inconsistent* user input, i. e., for which there may be no solution that would satisfy $\mathbf{o} = \mathbf{W}\mathbf{a}$. Therefore, we opt for a solution in the least squares sense $\mathbf{W}^{\mathrm{T}}\mathbf{o} = \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{a}$, which means that the *optimal* solution in our sense minimizes the quadratic function $||\mathbf{W}\mathbf{a}-\mathbf{o}||^2$.

Extinction describes the overall shape of a volume, but it is hard to optimize for. Emission and albedo are sufficient to change the appearance of an existing volume. Extinction cannot simultaneously be estimated in combination with emission or albedo, since it involves the solution of a complex non-linear problem, which is linearized by going into log-space. However, it is possible to begin with the reconstruction of extinction if L_{m} is constant. Hence, one can solve this simplified problem in a first step. Based on this result, one can then add scattering light and refine the appearance by estimating albedo and emission while lifting the constraint that L_{m} is constant.

4.4 Light Reconstruction

Alternatively to modifying the volume properties, one might also consider changing the environmental illumination. In the following, we explain how to add this additional optimization possibility, which aims at optimizing only the lighting of the scene in order to match the desired volume appearance as closely as possible. The volume properties emission, albedo and extinction are assumed to be spatially varying, but constant in this situation.

Light transport behaves linearly when employing the models typically used in rendering and one can formulate the complete light transport (including bounces) as a linear light transport operator T [42] applied to the outgoing radiance L_o coming from the environment. In a discrete setting, we can consider that L_o is a linear combination of a finite number M of basis functions that model the light sources and can, hence, be described as an M-dimensional coefficient vector $\vec{L}_{\mathrm{o}}$. Correspondingly, the transport operator is a matrix $\mathbf{T}$ of size $N \times M$, where N is the number of pixels in the images used as optimization constraints and, as before, the desired number of observation pixels $\mathbf{o}$ provided by the user. The condition that the volume properties are constant during this optimization implies that T is constant as well. Reconstruction then consists in determining an optimal outgoing radiance from the environment $\vec{L}_{\mathrm{o}}$ that best satisfies the following equation in a least-squares sense:

$$\mathbf{o} = \mathbf{T}\,\vec{L}_{\mathrm{o}}.$$

To determine $\mathbf{T}$, we need to evaluate Eq. 1. It is important to note, that $\vec{L}_{\mathrm{o}}$ represents only coefficients for predefined basis functions. In our case we consider directional light sources that represent a sampled environment map. But also other basis functions, e.g., describing local light sources could be used in the optimization framework. The previous equation is similar to the ones we derived earlier and, as we will show in the next section, we can apply a similar optimization scheme. Nonetheless, we will also see that a few additional steps need to be added in order to make the solution efficient (Sec. 5.3).

5 Implementation

In order to ensure a quick feedback to the user, we map our optimization to the GPU via compute shaders in OpenGL. We first describe the details for the optimization of the volume parameters (Sec. 5.1). We then introduce extensions to the method and additional acceleration details (Sec. 5.2). The light optimization (Sec. 5.3) is solved in a similar way, but requires some additional precomputation steps in order to speed up the optimization and to make it practical. In both cases, this mapping is not direct, as special care is needed regarding memory and multi-threading management.

5.1 Volume Reconstruction

The matrix $\mathbf{W}$ is large (total count of constraint pixels $\times$ number of basis functions), the linear system is ill-conditioned, and, finally, we seek a physically plausible, i. e. , a non-negative solution.

Fortunately, $\mathbf{W}$ is sparse because each row is derived by a ray passing through the volume, intersecting only a low number of basis functions v^i. Still it would be too much to keep all in memory. Instead, we implicitly solve the system by performing a conjugate gradient minimization [43] of the quadratic function $||\mathbf{W}\mathbf{a} - \mathbf{o}||^2$. All necessary steps of the conjugate gradient method are carried out on the GPU and involve 3D textures to represent the vectors. Operations on these vectors are implemented as shaders.

We employ the conjugate gradient method due to its fast convergence, but, for clarity, we illustrate the required operations by describing a standard gradient descent which yields the same minimum. The main ingredients are:

1) the computation of the gradient $\mathbf{W}^{\mathrm{T}}(\mathbf{W}\mathbf{a} - \mathbf{o})$,
2) an update of the current solution $\mathbf{a}$ by adding a scaled version of the gradient,
3) an iteration of the previous two steps.

Performing the update is straightforward, but the computation of the gradient is not.

To understand $\mathbf{W}^{\mathrm{T}}(\mathbf{W}\mathbf{a} - \mathbf{o})$, we examine its elements step by step. The matrix $\mathbf{W}$ encodes volume rendering (e.g., for Eq. (6)): $\mathbf{W}\mathbf{a} = \int_c T_r(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t) \left(\sum_i a^i\, v^i(\mathbf{x}_t) \right)$. Hence, $\mathbf{W}\mathbf{a}$ is determined by rendering a volume with properties $\mathbf{a}$, yielding multiple images. Next, computing $\mathbf{c} := (\ldots, c^k, \ldots)^{\mathrm{T}} := \mathbf{W}\mathbf{a} - \mathbf{o}$ is an image operation. Applying $\mathbf{W}^{\mathrm{T}}$ to $\mathbf{c}$ is the back-projection step that distributes the residual error over all voxels, a data scattering operation: Each value c^k is associated to a constraint pixel k. It needs to be scattered to those voxels that are traversed by ray-marching the ray associated with pixel k, weighted according to the voxel's influence on k. In other words, we perform a ray marching that is very similar to the rendering computation with $\mathbf{W}$.

In fact, both operations are so similar, that almost the same code is used. In rendering, which implements the multiplication by $\mathbf{W}$, we use ray-marching to sum the voxel contributions along the ray: $\mathrm{outRadiance} \mathrel{+}= \mathrm{weight} \cdot \mathrm{texRead}(\mathbf{a}, \mathrm{rayPos})$. The variable weight includes the transmittance between the current marching position rayPos and the ray origin due to the volumes' extinction, as well as the value of the basis function at the current position. To implement the multiplication by $\mathbf{W}^{\mathrm{T}}$, this single line is exchanged by $\mathrm{texWriteAdd}(\mathbf{a}, \mathrm{rayPos}, \mathrm{weight} \cdot c^k)$. Reusing the code is also beneficial as the weight values match up perfectly, and, consequently, the implicitly constructed matrices $\mathbf{W}$ and $\mathbf{W}^{\mathrm{T}}$ agree with each other. This is also reflected in the pseudo code in Listing 1.

```
void main() {
#ifdef RENDERING
  vec4 outputRadiance = vec4(0);
#elif defined(BACKPROJECT)
  vec4 backProj = texelFetch(texProj, pixelIdx);
#endif

  // setup view ray
  Ray ray = getRay(pixelIdx, cameraData);
  vec2 startEndRay = intersect(ray, volumeBBox);
  float dSurface = texRead(texDepth, pixelIdx).r;
  startEndRay = min(startEndRay, dSurface.xx);
  advanceRay(ray, enterExitVolume.x);

  vec4 sumExt = vec4(0);
  // perform ray marching
  for(int i=0;
    i < (startEndRay.y-startEndRay.x)/rayStepSize;
    ++i)
  {
    // account for extinction
    vec4 extinction = texRead(volExtinction, ray);
    vec4 weight = exp(-sumExt)*(1-exp(extinction));
#ifdef RENDERING
    outputRadiance += weight*
      texRead(volEmission, ray);
#elif defined(BACKPROJECT)
    texWriteAdd(volEmission, ray, weight*backProj);
#endif
    sumExt += extinction;
    advanceRay(ray, rayStepSize);
  }
#ifdef RENDERING
  texWrite(imgRendering, pixelIdx, outputRadiance);
#endif
}
```

Listing 1. Pseudo code: rendering and backprojection

5.2 Optimizations and Extensions

Data scattering performance can be improved by following a few observations. While multiplying with $\mathbf{W}$ is efficient and directly parallelized as usual when stepping through the volume per pixel, the situation is different for multiplication by $\mathbf{W}^{\mathrm{T}}$. As we scatter data (texture writes are realized via `shader_image_load_store`), synchronization issues may occur. Rays of neighboring pixels will likely write to the same voxel. To avoid the resulting stall, we use an interleaved pattern of 6×6 pixels. In each round, only one ray of these sub-windows is shot, decreasing the number of conflicts and speeding up the computation by $\approx 10\%$. Further, the use of 16bit instead of 32bit textures leads to a speedup of $\approx 25\%$ due to the reduced bandwidth.

Higher precision is obtained when using accurate ray traversal [44] instead of marching. For several basis-function choices, e.g., nearest-neighbor or linear (which we use), an accurate integral can be computed. This applies for the accumulated transmittance value during volume traversal as well as the weight of the basis function itself.

Regularization is a standard way of stabilizing and controlling the optimization. Usually, additional quadratic terms, modeling prior knowledge about

the solution space, are added to the quadratic error function to address numerical ill-conditioning. A Laplacian term $\|\mathrm{L}\mathbf{a}\|^2$ can smooth the overall volume. $\|\mathbf{a}\|^2$ minimizes the solution, while $\|\mathbf{a}-1\|^2$ biases it towards one. Each of these regularizers is controlled by user-defined weights. Thus, when optimizing for albedo and emission, we can specify which element to favor and, e.g., minimize emission. In practice, we use weak weights (i.e., 2^{-9}), which proved sufficient for a stable solution.

Constraint image weights are often desirable to give different parts of an image more importance than others. They are also handy when creating transitions, e.g., when editing only a part of the volume. These per-constraint pixel weights can be easily integrated into the conjugate gradient method and need to be multiplied with the vector $\mathbf{W}\mathbf{a}-\mathbf{o}$. An additional usage of weights is to steer the optimization process and favor certain parts. For example, one can consider putting more importance to the salient or important elements in the input images. The lowered weights give more room to the optimization of other views.

Visual Hulls can be used to limit the domain of solution [14], which increases quality and performance. In case of the emission/albedo optimization, we only consider voxels with non-zero extinction, as the other areas have no influence on the rendering. For estimating extinction itself, the input views of the user can be transformed into masks automatically, or, alternatively, the user can specify arbitrary masks (projections of the desired visual hull) as additional input to the optimization. Those masks do not need to align with any of the input images, the parts of the volume that are outside the visual hull are simply ignored during the optimization.

Refraction occurs for non-constant refractive indices and rays are bent during the traversal. Our reconstruction scheme can handle arbitrary integration curves c of known geometry (cf. Eq. 1), which enables us to use more complex paths than a straight line. To compute the refractive ray paths in our volumetric setting, we resort to the Euler forward scheme of [45].

Differing resolutions can be chosen for the different properties of the volume. In practice, albedo and emission can be of lower resolution without sacrificing too much quality. Hereby, computations are accelerated and memory usage is reduced. Although our software supports this possibility, we did not make use of it in the presented results.

5.3 Light Reconstruction

We have seen in the previous section that the optimization of the environmental lighting can also be described as the least-squares solution of a linear system: $\mathbf{o}=\mathbf{T}\,\vec{L}_\mathrm{o}$. As for the optimization of volume parameters, we will have to execute two matrix multiplications to converge to the optimal result; multiplying with $\mathbf{T}$ propagates the outgoing radiance $\vec{L}_\mathrm{o}$ to the constraint pixels, and multiplying with $\mathbf{T}^\mathrm{T}$ is a data scattering process to the light source coefficients. In other words, the light source coefficients take the role of the volume property to be optimized for and $\mathbf{T}$ replaces the matrix $\mathbf{W}$. While it is, hence, possible to directly execute the same scheme as for $\mathbf{W}$, such a direct solution would not be efficient.

The implicit construction of $\mathbf{T}$ is much more costly than it is for $\mathbf{W}$, so an on the fly evaluation is no option for acceptable performance. The reason is that $\mathbf{T}$ is no longer sparse - it encodes the complete light scattering in the scene. Consequently, we need precomputation strategies and store intermediate representations. We describe two different strategies; the first is to compute and store $\mathbf{T}$ entirely, but memory consumption grows linearly with the number of constraint pixels, the second strategy is slightly costlier during the optimization, but has a memory consumption linear in the size of the volume times the number of light coefficients. As in many cases, a few coefficients (around 16) are enough to capture the light's influence on the participating media sufficiently, the memory cost is manageable.

The first strategy works as follows. For each given light coefficient, we compute its influence on each constraint pixel, which corresponds directly to one column in $\mathbf{T}$. In other words, if all constraint pixels formed an image, we would simply render this image using an environmental lighting $\vec{L}_\mathrm{o}^i$ for which only the i^{th} coefficient is equal to one, all others zero. Each of these render steps will deliver one column of $\mathbf{T}$. In practice, any off-the-shelf renderer can be used, such as Optix [46]. During execution of the optimization steps, we can directly recover the needed coefficients from $\mathbf{T}$. This choice leads to a very efficient optimization, but $\mathbf{T}$ has size *light coefficients times constraint pixels*, which implies that memory consumption grows with the artistic input.

Alternatively, we propose to store *light volumes*; we determine the result of the light transport for each $\vec{L}_\mathrm{o}^i$ to each voxel in the volume. In other words, we light the voxel volume with $\vec{L}_\mathrm{o}^i$ and store the lit volume vol_i. To compute the influence of $\vec{L}_\mathrm{o}^i$ on a given constraint pixel, one can shoot the corresponding ray through the volume and gather the values from vol_i. Intuitively, this process is equivalent to associating each light-source coefficient to a light-source volume (the stored values can, in fact, be treated as an emission term). During the optimization, the multiplication with $\mathbf{T}$ is again a ray traversal, during which information can be collected from all vol_i simultaneously and stored in the constraint pixels. The multiplication with $\mathbf{T}^\mathrm{T}$ then has to distribute the residuals to the light coefficients, just like in the case of volume optimization, only that the values are stored in the volume, but the light coefficient vector.

We again encounter performance issues if we add

the residuals during the ray traversal directly to $\vec{L}_o^i$ in each step. Even when accumulating the result along the entire ray trajectory and adding the result only at the end to $\vec{L}_o^i$, we would need to use atomics as multiple rays will write to the same coefficient. Hence, many threads need to write to the same memory location, especially when using only a few light coefficients, causing serialization and low performance. We store these residuals with each ray and perform a mipmapping procedure to gather instead of scatter all ray contributions. The highest level of the mipmap pyramid then contains the values for $\vec{L}_o^i$.

The flexibility and independence of the artist's input, makes the second approach the preferred solution. While the memory consumption is significant (light coefficients times light volume resolution, for an isotropic phase function), it is usually acceptable. It would further be possible to reduce GPU memory usage if really needed. One strategy is to perform multiple ray traversals; during each traversal, only some vol_i are kept in the actual GPU memory, occupying only a small amount of memory at once. Nonetheless, exchanging the vol_i set is costly.

6 Results and Discussion

While the target images for the optimization can be arbitrary, a typical workflow is to render the volume with its current parameters (e.g., in the beginning with constant emission/albedo parameters) and modify the renderings to reflect the desired output. In all our examples we used Adobe Photoshop® to produce the desired target images. Other image editing tools, such as [47] or [48], would also be possible.

We ran the optimizations on a PC with an Intel x5650 and a nVidia 560Ti with fast execution times, even for medium sized voxel grids. For light optimization, the computation cost scales linearly with the number of light coefficients. A few are usually sufficient and lead to acceptable computation times.

For the volume optimization, the execution times for different numbers of ray marching steps, voxel resolution, and input images using the volume optimization are given in Fig. 4. In practice, twice the number of ray marching steps for the length of the diagonal as the number of voxels along an axis is a good choice. The process converges after three to five conjugate gradient steps. For 128^3 voxels, 256 ray marching steps, 460K constraint pixels (2 views), the result (Fig. 7) is computed in less than two seconds. With 2.3M constraint pixels (10 views) the computation takes seven seconds (Fig. 9).

For the light optimization, the execution times are given in Fig. 5. In general, good results can be achieved with a low number of light coefficients. Optimizing for 16 coefficients on a 128^3 volume with 2.3M constraint pixels (10 views), takes only around three seconds.

2 views 480x480 pixels each	volume resolution 32^3	64^3	128^3
ray marching steps / diagonal 64	47	65	143
128	64	86	168
256	98	126	217
512	164	207	317

steps/diag.	64	128	256
vol. resolution	32^3	64^3	128^3
# of views 480x480 pixels each 2	47	86	217
4	79	143	334
6	113	203	454
8	143	256	574

Fig. 4. Timings in ms per iteration (six for convergence) for the volume optimization of the hand example. Left: volume resolution vs. ray marching steps (relative to volume diagonal), more steps estimate the integrals more accurately. Right: volume resolution vs. # of views. The resolution of the volume was fixed to 128^3.

10 views, 640x360 pixels each; 256 ray marching steps (relative to volume diagonal)

light precomputation	light volume resolution 32^3	64^3	128^3
light coeffs. 16	0.2	0.2	0.27
64	0.78	0.81	1.07
256	3.72	3.75	5.39

optimization iteration	light volume resolution 32^3	64^3	128^3
light coeffs. 16	0.62	0.63	0.63
64	2.39	2.44	2.46
256	10.28	10.28	10.52

memory consumption	light volume resolution 32^3	64^3	128^3
light coeffs. 16	2	16	128
64	8	64	512
256	32	256	2048

Fig. 5. Timings in seconds / memory consumption in MB for City data set (10 views, each 640×360 pixels). The volume resolution of the underlying extinction volume was fixed to 128^3 and, correspondingly, the number of ray marching steps set to 256. The tables show number of light coefficients vs. light volume resolution. Left: time for pre-computation of light volumes. Middle: time for a single optimization iteration (six for convergence). Right: memory consumption of the light volumes. The overall speed scales directly with the number of light coefficients. The light volume resolution has a minor impact on the performance as it is further dominated by the number of ray marching steps, which need to correspond to the resolution of the underlying extinction volume. This behavior is equal to the volume optimization.

In the following, we illustrate our method on several examples and describe the intent of each of the stylizations. It took a user roughly 2 minutes to produce the smoke hand, 20 min for the volcano, 15 min for the city cloud, and 4 min for the refraction.

Smoke Hand

shows a particular case where the goal is to add imprinted logos in the volume. These were added to the user-provided views. When optimizing albedo and emission, the result is a close match to the input (Fig. 7, top) and intermediate views look appealing and plausible (middle). One can also modify emission or albedo only (bottom), in which case the result deviates from the input and can no longer match it perfectly. Emission can only lighten, albedo only darken the appearance (following physical constraints). Depending on the realism and desired material, these could be adequate choices as well.

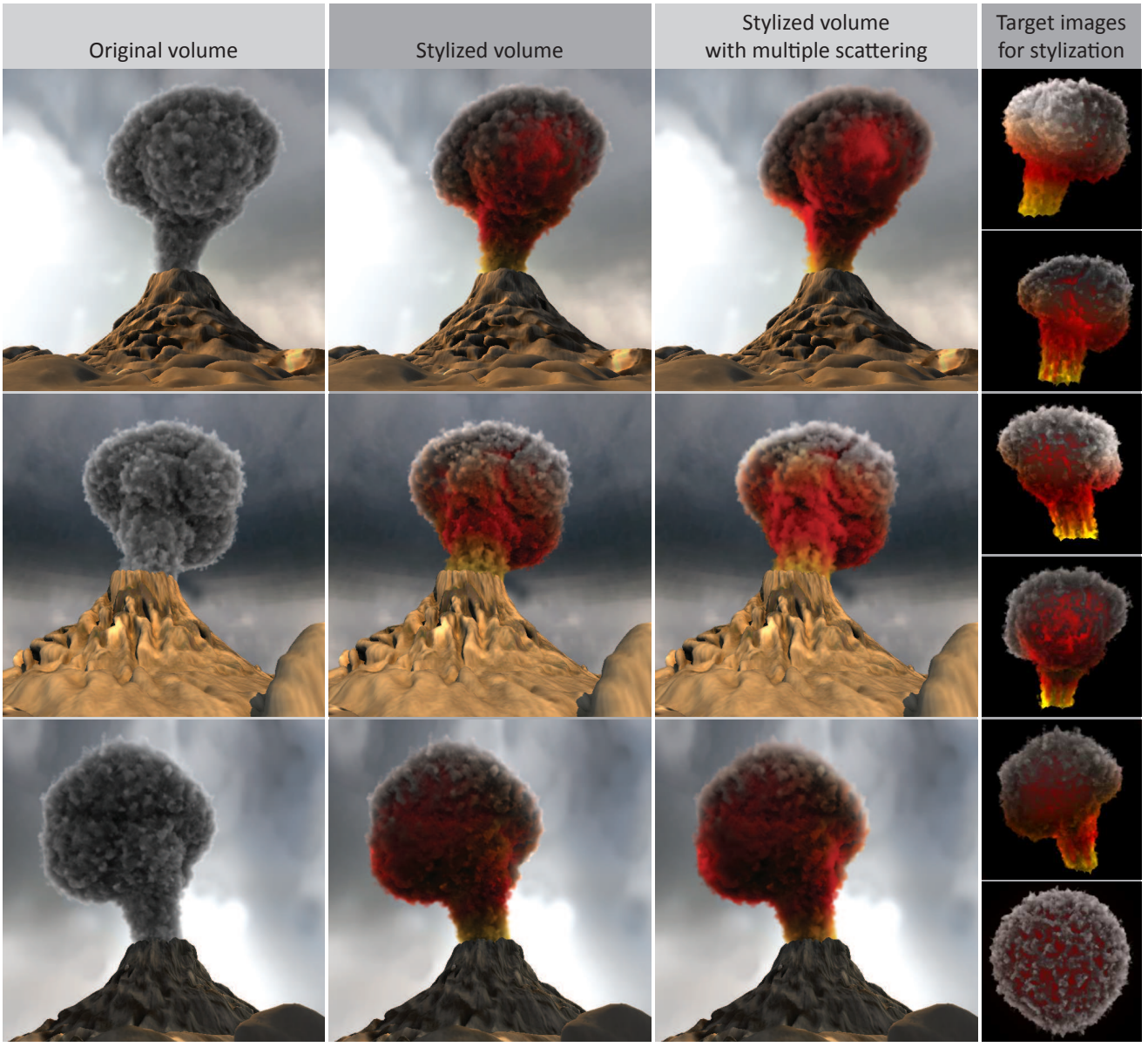

Fig. 6. Stylization of the static smoke of a volcano; original volume (left), intermediate views of the result after optimization (middle), rendering with multiple scattering (unlimited bounces) of the optimized volume (right). While we optimize only for single-scattering, the results remain suitable for a full light transport simulation, leading only to a minor loss in contrast. To account for the additional energy, we globally scaled the emission and incoming light such that the resulting images have similar brightness. The right-most column shows all user-drawn images.

Volcano Stylization

demonstrates the effectiveness of our system, using the volume optimization. The scene shows a static smoke cloud from a volcano (Fig. 6). The stylization intends to make the volcano appear more active with ongoing dynamics.

Six input images (as for the faces of a cube) were used (480×480 pixel, Fig. 6, right-most column) to change the appearance of the smoke, resulting in appealing renderings from any viewing angle. A lower number of target images is insufficient for a good appearance all around the object. In general, only parts that are specified in at least a single view change, all other parts (e.g., smoke inside the volcano, occluded by the volcano itself) remain unaltered. Consequently, edges can appear between constrained and unconstrained regions. Also, as constraint pixels affect only voxels along their corresponding ray, insufficient views may induce stripe patterns into the volume. However, applying a weak Laplacian regularizer introduces diffusion between neighboring voxels, reducing both, stripe and edge artifacts greatly without smoothing too much of the intricate detail.

The effect of rendering the stylized volume under

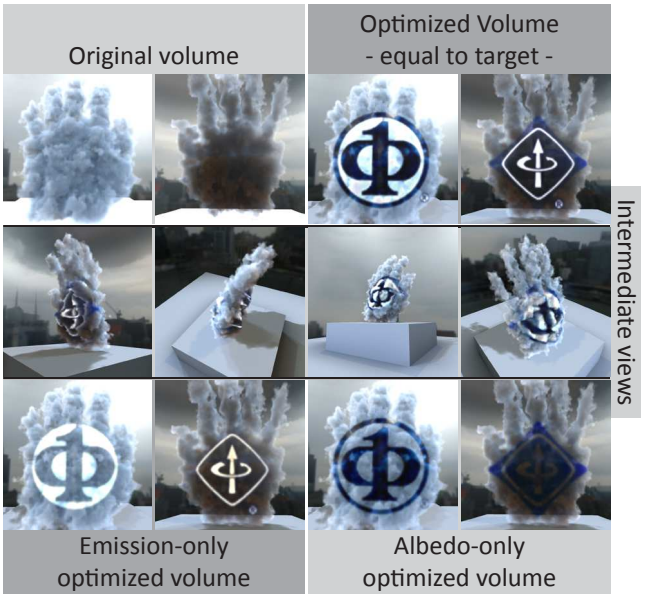

Fig. 7. Logos on a smoke hand. Top: original appearance next to the optimized result (emission and albedo). The images of the reconstruction are indistinguishable from the desired input views. Middle: Intermediate views appear plausible. Bottom: left two images use only emission, hence cannot fully match the desired appearance as light is only added. The other two use only albedo, this time, light is only removed.

multiple scattering is shown in the right column of Fig. 6. The appearance remains pleasing, although multiple scattering was not included in our optimization process. It acts like a blur, that reduces details, but the resulting rendering remains consistent.

Extinction and Refraction

is shown in Fig. 8. Here, we illustrate the use of our extinction optimization to show the construction of complex shapes and also to illustrate the compatibility of our solution with bent rays due to continuous refraction. The user provided four input views (growth of a tree) and our reconstruction computed extinction coefficients (per wavelength) that attenuate the background light such as to match the provided images. In the example, the volume does not scatter or emit any light, i. e., $L_{\mathrm{m}} = 0$. In this reconstruction process, the shape of the volume is implicitly defined by the volume of varying refractive index. The refraction indices were generated using a random process restricted to the inside of a glass sphere. We used per-pixel weights to concentrate the importance on the main parts of the user images, which leads to a less constrained boundary and a more variable shape. Due to the refraction, the intermediate views appear randomly colored. We imagine, such a combination of refraction and extinction could e.g. be used in a game, where the user has to find the correct viewpoint to

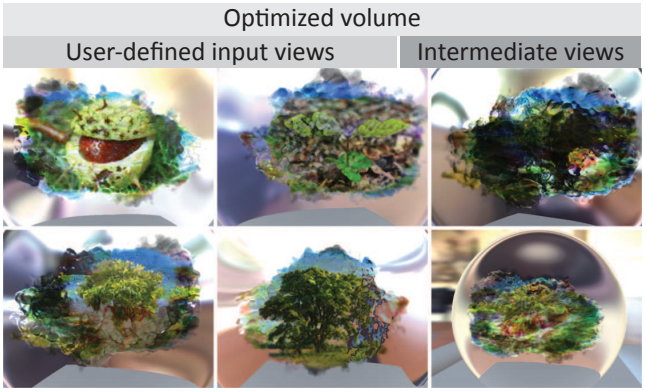

Fig. 8. Reconstruction of extinction in a refractive volume to absorb light of the background. Left, middle: results for user-defined views, rightmost column: intermediate views.

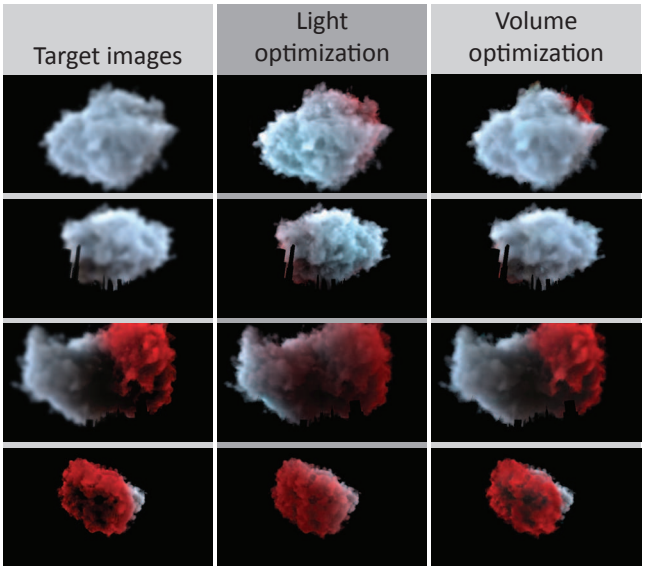

Fig. 9. Comparison of the reconstruction results (subtracted background): user-drawn input images (left), results after light optimization (middle), results after volume optimization (right).

see a certain image, in order to obtain hints to solve a puzzle.

Cloud Stylization

illustrates the expressiveness of our system. The scene shows a cloud over a city and the goal is to achieve a "frightening" look when on one side of the cloud, a calm appearance on the other side.

The user modified ten views (640×360 pixel, a selection is shown in Fig. 9, left) by contrast enhancement and blending with a red mask. The resulting input images were *inconsistent* in parts, as each was independently designed. Yet, our least-squares solutions ensure a valid reconstruction that closely matches the desired scene appearance and extrapolates well.

Fig. 10. Pair-wise comparison: applying volume optimization (right in pair) on the result of light optimization (left in pair) can better match the desired appearance, using the same target images. Contrast can be locally decreased (top), but also increased (middle, bottom) according to the target images.

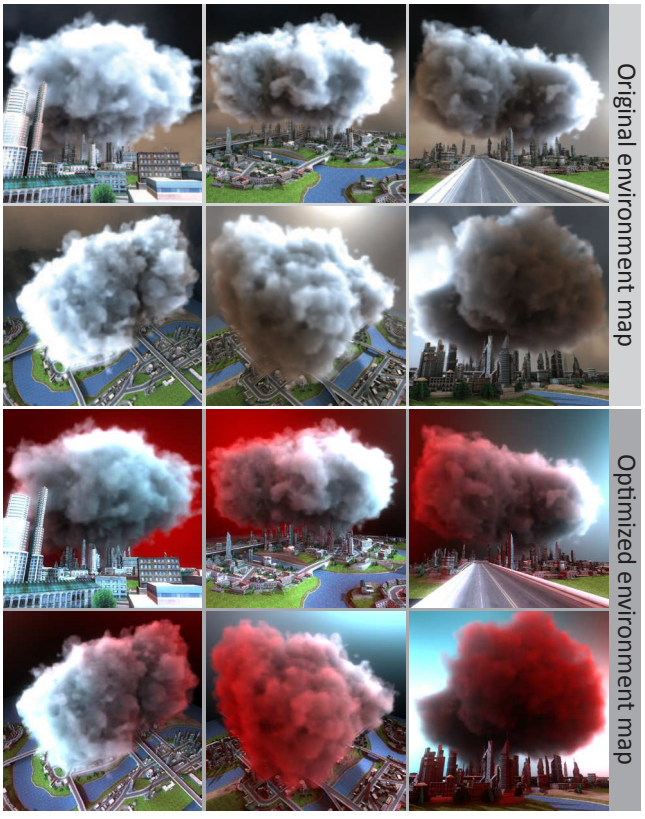

Fig. 11. Cloud Stylization via environmental lighting. The volume's original appearance (top) is changed due to a new environment map (bottom). The views are different from the target images.

We applied two different optimization strategies. First, we modified the environmental illumination (Fig. 11). As light basis functions, we used 128 uniformly-distributed disc lights restricted to the upper hemisphere that cover a constant solid angle with a soft falloff. The resulting environment map (all light sources splat into the image) is illustrated in Fig. 12.

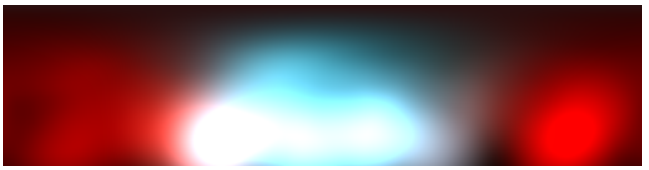

Fig. 12. Optimized environment map in latitude-longitude format. The lower half is omitted as the center directions of the light sources were chosen to be on the upper hemisphere only.

It is important to notice that an isotropic volume acts as a low-pass filter; from the light source to the camera, as well as from the camera to the light sources. Hence, the matching to the target images cannot be perfect. High frequency information, such as the logos in Fig. 7, cannot be reproduced in this case. Consequently, the resulting environmental lighting is also of a low frequency, justifying the low number of light coefficients. In practice, as few as 16 coefficients are often sufficient.

The volume optimization on the city data set, is able to reconstruct higher frequencies because the volume is modified locally (Fig. 9). In fact, when allowing for emission, light optimization is theoretically redundant. It can be useful to include, e.g. when certain volume properties are not supposed to change or if emission is to be avoided. It is also a good means for rapid preview, as even a single view can be used without introducing any potential high-frequency artifacts stemming from a very low number of input images during the property reconstruction. Further, optimizing for light has a more global character as each light source affects the entire volume, which may be desired. It naturally connects the appearance of the volume to the rest of the scene, when lit with the same environmental illumination (Fig. 10).

Animation

While our optimization targets static volumes, a solution for animated and dynamic volumes would be of interest. Here, we analyze a simple extension, but we leave a full solution of this problem to future work.

If the volumetric data set is static, one way of incorporating a changing environment (e.g., light conditions and time-based constraints) would be to find the best solution (in a least-squares sense) over time, which can be achieved by our approach by extending the constraint system.

The more interesting case are dynamic volumes. Here, a straightforward solution is to use keyframing and optimization for multiple volumes at different time steps that are linearly blended together to produce in-between frames. Yet, this simple solution can lead to problems; if the extinction is zero at a location in two neighboring key frames, only random

values are derived at these locations. If the volume's animation now passes through this region, the result could be arbitrary. Using the Laplace regularizer as a diffusion process, can solve this issue, as it extends the emission/albedo definitions into regions with zero extinction. Please refer to the video for an example.

A more advanced optimization of dynamic volumes, requires an understanding of the volume's underlying dynamics and is beyond the scope of this paper. One could, e.g., couple the optimized emission/albedo volume and the animated extinction volume by a flow field. The latter could directly come from a fluid simulation or be a reconstructed volumetric flow. Hereby, one could link voxel values over time, according to the flow. Currently, we can only add such constraints to the same voxel over time, making the definition of key frames only slightly more flexible. The additional flow information would allow more expressive and appealing results, but it would also lead to a computational and memory overhead.

7 Conclusions

This paper presents a novel approach to stylize static volumes. We show that a desired scene appearance can be defined via a number of target views that our approach then optimizes for. We consider two scenarios to solve for a specific volume appearance: the modification of volume properties and the modification of environmental illumination. In both cases, we show how to formalize the task as a linear optimization problem. The optimization depends on the solution of a large linear system. Its solution is made practical by employing an efficient GPU-friendly implementation that avoids the explicit construction of the system. Further, we illustrate the usefulness, expressiveness, and variety of our system with a collection of different examples.

In the future, multiple scattering could be interesting, but it results in a non-linear optimization. It comes with two additional challenges, how to quickly project a volume after parameters have been selected and how to back project the residual. A potential avenue could be a combination of voxel-based out-of-core rendering [49] and approximate indirect illumination[50]. Finally, we made a small step in the direction of animation, but fully dynamic volumes remain a challenge.

Acknowledgements

We would like to thank Bernhard Reinert, and colleagues at MPI for helpful discussions and the reviewers for their valuable feedback. This work was partly supported by the German Research Foundation (DFG) through the Emmy-Noether fellowship IH 114/1-1, by the European Union via the FP7 FET Open Harvest4D, and by the Intel Visual Computing Institute at Saarland University. The city model is from 3DRT.

References

[1] J. Kajiya and B. Von Herzen, "Ray tracing volume densities," in *Proc. ACM SIGGRAPH*, 1984, pp. 165–174.
[2] N. Max, "Optical models for direct volume rendering," *IEEE TVCG*, vol. 1, no. 2, pp. 99–108, 1995.
[3] W. Jarosz, D. Nowrouzezahrai, R. Thomas, P.-P. Sloan, and M. Zwicker, "Progressive photon beams," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 181:1–181:12, 2011.
[4] J. Novák, D. Nowrouzezahrai, C. Dachsbacher, and W. Jarosz, "Progressive virtual beam lights," *Computer Graphics Forum*, vol. 31, no. 4, pp. 1407–1413, 2012.
[5] J. Stam, "Stable fluids," in *Proc. ACM SIGGRAPH*, 1999, pp. 121–128.
[6] K. Perlin, "Hypertexture," in *Proc. ACM SIGGRAPH*, 1989, pp. 253–262.
[7] A. Treuille, A. McNamara, Z. Popović, and J. Stam, "Keyframe control of smoke simulations," in *Proc. ACM SIGGRAPH*, 2003, pp. 716–723.
[8] A. McNamara, A. Treuille, Z. Popović, and J. Stam, "Fluid control using the adjoint method," in *Proc. ACM SIGGRAPH*, 2004, pp. 449–456.
[9] J. Obert, J. Kriványek, F. Pellacini, D. Sýkora, and S. N. Pattanaik, "icheat: A representation for artistic control of indirect cinematic lighting," *Comput. Graph. Forum*, vol. 27, no. 4, pp. 1217–1223, 2008.
[10] W. B. Kerr, F. Pellacini, and J. D. Denning, "Bendylights: Artistic control of direct illumination by curving light rays," *Comput. Graph. Forum*, vol. 29, no. 4, pp. 1451–1459, 2010.
[11] D. Nowrouzezahrai, J. Johnson, A. Selle, D. Lacewell, M. Kaschalk, and W. Jarosz, "A programmable system for artistic volumetric lighting," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 29:1–29:8, 2011.
[12] O. Klehm, I. Ihrke, H.-P. Seidel, and E. Eisemann, "Volume stylizer: Tomography-based volume painting," in *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, 2013, pp. 161–168.
[13] C. De Rousiers, A. Bousseau, K. Subr, N. Holzschuch, and R. Ramamoorthi, "Real-time rough refraction," in *Proc. ACM SIGGRAPH i3D*, 2011, pp. 111–118.
[14] I. Ihrke and M. Magnor, "Image-based tomographic reconstruction of flames," *Proc. SCA*, pp. 367–375, 2004.
[15] T. Hawkins, P. Einarsson, and P. Debevec, "Acquisition of time-varying participating media," in *Proc. ACM SIGGRAPH*, 2005, pp. 812–815.
[16] I. Ihrke and M. Magnor, "Adaptive grid optical tomography," *Graphical Models*, vol. 68, pp. 484–495, 2006.
[17] I. Ihrke, B. Goldluecke, and M. Magnor, "Reconstructing the geometry of flowing water," in *Proc. ICCV*, 2005, pp. 1055–1060.
[18] B. Atcheson, I. Ihrke, W. Heidrich, A. Tevs, D. Bradley, M. Magnor, and H.-P. Seidel, "Time-resolved 3d capture of non-stationary gas flows," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 132:1–132:9, 2008.
[19] A. Linţu, H. P. A. Lensch, M. Magnor, S. El-Abed, and H.-P. Seidel, "3d reconstruction of emission and absorption in planetary nebulae," in *Proc. Volume Graphics*, 2007, pp. 9–16.
[20] C. Schoeneman, J. Dorsey, B. E. Smits, J. Arvo, and D. Greenburg, "Painting with light," in *Proc. ACM SIGGRAPH*, 1993, pp. 143–146.
[21] J. Obert, F. Pellacini, and S. N. Pattanaik, "Visibility editing for all-frequency shadow design," *Comput. Graph. Forum*, vol. 29, no. 4, pp. 1441–1449, 2010.
[22] Y. Song, X. Tong, F. Pellacini, and P. Peers, "Subedit: a representation for editing measured heterogeneous subsurface scattering," *ACM Trans. Graph.*, vol. 30, no. 3, pp. 31:1–31:10, 2009.
[23] S. Chandrasekar, *Radiative Transfer*. Dover Pub., 1960.
[24] M. Hašan and R. Ramamoorthi, "Interactive albedo editing in path-traced volumetric materials," *ACM Trans. Graph.*, vol. 32, no. 2, pp. 11:1–11:11, 2013.
[25] T. Ropinski, J. Praßni, F. Steinicke, and K. Hinrichs, "Stroke-based transfer function design," in *IEEE/ EG Symposium on Volume and Point-Based Graphics*, 2008, pp. 41–48.
[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. Nguyen, T. Ritschel, K. Myszkowski, E. Eisemann, and H.-P. Seidel, "3D material style transfer," *Computer Graphics Forum (Proc. Eurographics)*, vol. 31, no. 2, pp. 431–438, 2012.

[28] Y. Dobashi, W. Iwasaki, A. Ono, T. Yamamoto, Y. Yue, and T. Nishita, "An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 145:1–145:10, 2012.

[29] M. Okabe, Y. Matsushita, L. Shen, and T. Igarashi, "Illumination brush: Interactive design of all-frequency lighting," in *Proc. Pacific Graphics*, 2007, pp. 171–180.

[30] F. Pellacini, F. Battaglia, R. K. Morley, and A. Finkelstein, "Lighting with paint," *ACM Trans. Graph.*, vol. 26, no. 2, pp. 9:1–9:14, 2007.

[31] M. Pauly and N. Mitra, "Shadow art," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 156:1–156:7, 2009.

[32] I. Baran, P. Keller, D. Bradley, S. Coros, W. Jarosz, D. Nowrouzezahrai, and M. Gross, "Manufacturing layered attenuators for multiple prescribed shadow images," *Computer Graphics Forum*, vol. 31, no. 2, pp. 603–610, 2012.

[33] M. Papas, W. Jarosz, W. Jakob, S. Rusinkiewicz, W. Matusik, and T. Weyrich, "Goal-based caustics," *Computer Graphics Forum*, vol. 30, no. 2, pp. 503–511, 2011.

[34] M. Papas, T. Houit, D. Nowrouzezahrai, M. Gross, and W. Jarosz, "The magic lens: Refractive steganography," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 186:1–186:10, 2012.

[35] T. Weyrich, P. Peers, W. Matusik, and S. Rusinkiewicz, "Fabricating microgeometry for custom surface reflectance," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 32:1–32:6, 2009.

[36] D. Lanman, G. Wetzstein, M. Hirsch, W. Heidrich, and R. Raskar, "Polarization fields: Dynamic light field display using multi-layer LCDs," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 186:1–186:10, 2011.

[37] G. Wetzstein, D. Lanman, M. Hirsch, and R. Raskar, "Tensor displays: Compressive light field synthesis using multilayer displays with directional backlighting," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 80:1–80:11, 2012.

[38] M. Fuchs, R. Raskar, H.-P. Seidel, and H. P. A. Lensch, "Towards passive 6d reflectance field displays," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 58:1–58:8, 2008.

[39] M. Holroyd, I. Baran, J. Lawrence, and W. Matusik, "Computing and fabricating multilayer models," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 187:1–187:8, 2011.

[40] G. Wetzstein, D. Lanman, W. Heidrich, and R. Raskar, "Layered 3d: Tomographic image synthesis for attenuation-based light field and high dynamic range displays," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 95:1–95:12, 2011.

[41] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[42] E. Veach, "Robust monte carlo methods for light transport simulation," Ph.D. dissertation, Stanford University, 1997.

[43] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep., 1994.

[44] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *Proc. Eurographics*, 1987, pp. 3–10.

[45] I. Ihrke, G. Ziegler, A. Tevs, C. Theobalt, M. Magnor, and H.-P. Seidel, "Eikonal rendering: Efficient light transport in refractive objects," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 59:1–59:8, 2007.

[46] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "Optix: a general purpose ray tracing engine," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 66:1–66:13, 2010.

[47] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Proc. ACM SIGGRAPH*, 2004, pp. 689–694.

[48] L.-Q. Ma and K. Xu, "Vea 2012: Efficient antialiased edit propagation for images and videos," *Comput. Graph.*, vol. 36, no. 8, pp. 1005–1012, 2012.

[49] C. Crassin, F. Neyret, M. Sainz, and E. Eisemann, *GPU Pro*. AK Peters, 2010, ch. X.3 Efficient Rendering of Highly Detailed Volumetric Scenes with GigaVoxels, pp. 643–676.

[50] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive indirect illumination using voxel cone tracing," *Computer Graphics Forum*, vol. 30, no. 7, pp. 1921–1930, 2011.

Oliver Klehm is a PhD student in the computer graphics group at the Max Planck Institute (MPI) for Informatics. He received a MSc degree from the Hasso-Plattner-Institut, Potsdam, Germany in 2011. His research interests include interactive and offline global illumination and visibility algorithms, as well as general-purpose GPU programming.

Ivo Ihrke is a permanent researcher at Inria Bordeaux Sud-Ouest where he leads the research group "Generalized Image Acquisition and Analysis" which is also supported by an Emmy-Noether fellowship of the German Research Foundation (DFG). Prior to that he was heading a research group within the Cluster of Excellence "Multimodal Computing and Interaction" at Saarland University. He was an Associate Senior Researcher at the MPI Informatik, and associated with the Max-Planck Center for Visual Computing and Communications. Before joining Saarland University he was a postdoctoral research fellow at the University of British Columbia, Vancouver, Canada, supported by the Alexander von Humboldt-Foundation. He received a MS degree in Scientific Computing from the Royal Institute of Technology (KTH), Stockholm, Sweden (2002) and a PhD (summa cum laude) in Computer Science from Saarland University (2007).

Hans-Peter Seidel is the scientific director and chair of the computer graphics group at the Max Planck Institute (MPI) for Informatics and a professor of computer science at Saarland University, Saarbrücken, Germany. He is co-chair of the Max Planck Center for Visual Computing and Communication (MPC-VCC) (since 2003), and he is the scientific coordinator of the Cluster of Excellence on Multimodal Computing and Interaction (M2CI) that was established by the German Research Foundation (DFG) within the framework of the German Excellence Initiative in 2007. In addition, Seidel is a member of the Governance Board of the newly established Intel Visual Computing Institute (IVCI) (since 2009). Seidel is a recipient of the DFG Leibnitz-Prize (2003) and of the Eurographics Distinguished Career Award (2012).

Elmar Eisemann is a Professor in Computer Science, heading the Computer Graphics and Visualization Section at Delft University of Technology. Before, he was Associate Professor at Télécom ParisTech (ENST) until 2012 and a senior researcher and group leader in the Cluster of Excellence at the Max-Planck-Institute and Saarland University (2008-2009). He studied at the École Normale Suprieure Paris (2001-2005) and completed his PhD at INRIA Rhône-Alpes (2005-2008). He is an author of the book "Real-Time Shadows" (AK Peters) and co-organized EGSR 2010, 2012, and HPG 2012. In 2011, he received the Eurographics Young Researcher Award.