Real-time Light Transport in Analytically Integrable Quasi-heterogeneous Media

Tomáš Iser*
Supervised by: Oskar Elek†

Faculty of Mathematics and Physics
Charles University
Prague / Czech Republic

Abstract

Our focus is on the real-time rendering of large-scale volumetric participating media, such as fog. Since a physically correct simulation of light transport in such media is inherently difficult, the existing real-time approaches are typically based on low-order scattering approximations or only consider homogeneous media.

We present an improved image-space method for computing light transport within quasi-heterogeneous, optically thin media. Our approach is based on a physically plausible formulation of the image-space scattering kernel and analytically integrable medium density functions. In particular, we propose a novel, hierarchical anisotropic filtering technique tailored to the target environments with inhomogeneous media. Our parallelizable solution enables us to render visually convincing, temporally coherent animations with fog-like media in real time, in a bounded time of only milliseconds per frame.

Keywords: Real-time Rendering, Light Transport, Multiple Scattering, Participating Media, GPU Shaders

Figure 1: Typical outputs rendered with our approach in different demo scenes. Both scenes contain several point light sources and emissive textures. They are blurred in an inhomogeneous fog with an altitude-dependent density.

1 Introduction

In computer graphics, one of our major topics is generating physically plausible images. This process is usually referred to as photorealistic rendering. It can be used in architectural or design visualizations, simulators, video games, or movies. The goal is to create imagery that resembles reality as we perceive it.

To achieve photorealistic results, it is necessary to understand and simulate how light interacts with the matter around us. When rendering simple scenes, the physical processes can be simplified by assuming that light travels in a vacuum. But once we want to render more advanced optical phenomena, such as *light scattering*, one can no longer rely on simplifications valid only in a vacuum.

Light scattering is a process where the otherwise straight trajectory of the light can be deviated, which happens when light travels through air, water, milk, and other non-metallic participating media [3]. Numerous optical phenomena, such as the blurring of objects and lights in a foggy weather, can be explained by the scattering.

Existing Approaches In the case of non-real-time rendering, there exist physically precise, but rather slow methods. Efficient Monte Carlo approaches are described, for example, by Jarosz [4]. However, rendering of a single picture may take several hours or even days. Furthermore, even a slight adjustment of the medium parameters can significantly change the rendering time. When we need our visualizations to run in *real time*, such as in video games or simulations, we need very efficient approxima-

*tomasiser@gmail.com
†oskar.elek@gmail.com

tions to process a single frame in only a few milliseconds.

Current real-time approaches, which we compare later in Section 6, are mostly based on empirical or single-scattering approximations [6, 8, 14]. These solutions are very fast, but ignore multiple scattering of light, which is reasonable only in optically very thin media. They typically compute in-scattering from light sources only, disregarding emissive materials and reflections. Therefore, they cannot blur the geometry in the scenes, which is an important effect noticeable in real media. Multiple-scattering real-time solutions exist only for homogeneous media [3] or we found them too slow for real-time use [12].

Our Approach In this paper, we aim at rendering scenes that are *filled with the medium* (Figure 1), i.e., the camera itself is in the medium. This corresponds to various real-life situations: foggy weather, sandstorms, when looking at a mist above a lake, or even when swimming in the lake itself. Our approach that we propose and implement is based on novel modifications of a solution originally described by Elek et al. [3] for homogeneous media only. The method we propose is capable of simulating *multiple-scattering* effects in *inhomogeneous media* in *real time*. In particular, our key achievements in this paper are:

- We propose a novel *multiple-scattering* approximation suitable for *quasi-heterogeneous* media and introduce closed-form solutions for the necessary density integrals (Section 3).
- We use efficient *hierarchical screen-space filtering*: we use the original filtering of Elek et al. [3] and extend it for inhomogeneous media and by adding new steps for *visual improvements* (Section 4).
- We show results indicating that our method is indeed capable of being executed *in real time* with a *stable framerate* even in Full HD resolutions (Section 5).

2 Fundamentals

In order to describe our approach in detail, we first briefly explain screen-space approaches in general, and then the physics of light transport in participating media.

2.1 Screen-space Methods

In this paper, we present a *screen-space* method [5, 3] intended to be used in a post-processing step, i.e., applied on an original frame unaffected by any participating medium. We treat *every single pixel* of the original frame as an implicit light source, as a source of *radiance L* incoming to the camera from a surface on that pixel. Intuitively, the pixels represent packets of photons traveling from surfaces towards the camera. By adding a participating medium, these photons (pixels) are affected, so we need to *post process* the individual pixels of the original frame.

Screen-space methods are popular and beneficial as their time complexity only depends on the number of pixels, i.e., the resolution, and *not* on the complexity of the rendered scene. Apart from participating media rendering [3, 14, 12], they are used for various other effects such as *depth of field* rendering introduced by Lee et al. [5].

2.2 Participating Media

For our purposes, we model a *participating medium* as a collection of identical, randomly positioned, and small particles, such as water droplets or dust. It allows us to describe media by the following properties, explained in more details by Elek [2] and Jarosz [4].

Photons travelling through such media can collide with the particles in the media. When such an interaction occurs, the photon itself can be completely *absorbed*, or it can be *scattered* in a new direction. Furthermore, certain media emit new photons, which is called *emission*.

For the position $\mathbf{x}$ in a certain medium, let $\rho(\mathbf{x})\left[\mathrm{m}^{-3}\right]$ denote the density of the particles at that location. Each particle is characterized by three properties: absorption cross-section $C_\mathrm{a}\left[\mathrm{m}^2\right]$, scattering cross-section $C_\mathrm{s}\left[\mathrm{m}^2\right]$, and a phase function f_p. Both C_a and C_s are *color-channel-dependent*, which will be implicit throughout the rest of the paper and is taken into account in our implementation.

Absorption Absorption happens when a photon is *absorbed* by a particle it hit. We consider the light energy to be "lost". Macroscopically, the "lost energy" can be observed as a decrease of the overall intensity of the light. How much the medium absorbs light is determined by the *absorption coefficient* $\sigma_\mathrm{a}\left[\mathrm{m}^{-1}\right]$: $\sigma_\mathrm{a}(\mathbf{x}) = C_\mathrm{a} \cdot \rho(\mathbf{x})$.

Scattering If a photon is not absorbed when it hits a particle, its energy is scattered into a new direction. The new direction depends on the *phase function* f_p of the medium, which we can understand as a probability distribution function. How much the scattering happens is described by the *scattering coefficient* $\sigma_\mathrm{s}\left[\mathrm{m}^{-1}\right]$: $\sigma_\mathrm{s}(\mathbf{x}) = C_\mathrm{s} \cdot \rho(\mathbf{x})$.

In a beam of light passing through a medium, the photons can be scattered *out* of the path (*out-scattering*), and other photons can converge *into* the beam (*in-scattering*). Furthermore, we distinguish between *single scattering*, when a single photon can only scatter once, and *multiple scattering*. Multiple scattering causes *spatial spreading* (Figure 2) and *angular spreading*, which can be observed in real media, e.g., as blurring of distant objects in a fog.

Optical Thickness *Extinction* describes when a photon is either absorbed or scattered. We define the *extinction coefficient* as $\sigma_\mathrm{t}\left[\mathrm{m}^{-1}\right] = \sigma_\mathrm{a} + \sigma_\mathrm{s}$. By integrating the extinction along a line segment l, we get the *optical thickness* (*optical depth*): $\tau(l) = \int_l \sigma_\mathrm{t}(\mathbf{x})\,\mathrm{d}\mathbf{x}$.

Beer–Lambert Law The optical thickness is related to *transmittance* T, where $T(l) = T(\mathbf{x_0}, \mathbf{x_1})$ expresses the radiance between two points $\mathbf{x_0}, \mathbf{x_1}$ after being reduced by interacting with the medium: $L(\mathbf{x_1}, \omega) = T(l) \cdot L(\mathbf{x_0}, \omega)$. According to *Beer–Lambert law*, the relation between the transmittance and the optical thickness along l is exponential: $T(l) = \exp(-\tau(l))$.

3 Our Scattering Approximation

In this section, we propose a novel *multiple-scattering* approximation suitable for inhomogeneous media. We first show an existing approach for homogeneous media (Section 3.1). We then propose and derive our novel solution for *quasi-heterogeneous media* (Section 3.2).

3.1 Spreading in Homogeneous Media

We can think of light scattering as a stochastic process with many realizations. The central limit theorem allows us to assume the spatial distribution of radiance to be Gaussian (Figure 2). This approach was used by Premože et al. [9] in their multiple-scattering approximations.

For a collimated pencil of light between two points with the distance s in a homogeneous medium, Premože et al. derived a formula for the *standard deviation* W of the Gaussian spatial-spreading distribution:

$$W(s) = \sqrt{\frac{1}{2}\left(\frac{2\sigma_\mathrm{a}}{3s} + \frac{4}{s^3\sigma_\mathrm{s}(1-g)}\right)^{-1}}, \qquad (1)$$

where $g \in [-1,1]$ is the *scattering asymmetry factor* describing the directions in which the photons scatter (e.g., for $g > 0$, scattering in forward directions is favored).

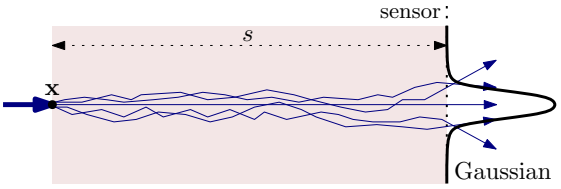

Figure 2: Spatial spreading of radiance caused by multiple scattering. We approximate the spreading width detected on the sensor (camera) to have a Gaussian distribution.

3.2 Proposed Quasi-heterogeneous Media

Unfortunately, Eq. 1 assumes the medium parameters to be spatially invariant along a light path. We propose a concept of *quasi-heterogeneous media* with a smooth density function $\rho(\mathbf{x})$. The density is therefore similar in a local neighborhood of a light ray, so we do not need a complicated anisotropic kernel. It remains to explain how we propose to take the spatially-varying density into account.

Idea: Density-dependent Distance Scaling When light travels along a line segment with the length D, the spreading width $W(D)$ depends on the scattering interactions occurring along the segment. By looking at the light scattering as a stochastic process, we propose to assume that $W(D)$ does not depend on where exactly the particles are accumulated along the line segment. We are only interested in how many particles along the path *in total* could have interacted with the light.

When a pencil of light collides with a higher number of particles, it spreads more than when it collides with less particles, even if the distance D is the same. Hence we propose to scale the distances w.r.t. the density *accumulated along the line segments*, which corresponds to *integrating the density* along the segment.

Mathematical Derivation We want to find a new length D' of a line segment l in a homogeneous medium that would "match" the behavior in a quasi-heterogeneous medium. Assume a homogeneous medium with a constant density ρ' and an optical thickness $\tau(l) = \int_l \sigma_\mathrm{t}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sigma_\mathrm{t} \cdot D$. The density of the quasi-heterogeneous medium is $\rho(\mathbf{x}) = \rho'\rho''(\mathbf{x})$, where $\rho''(\mathbf{x})$ is an arbitrary spatially-varying ratio between the homogeneous and heterogeneous density. The optical thickness in the quasi-heterogeneous medium is by definition $\tau(l) = \int_l \sigma_\mathrm{t}(\mathbf{x})\,\mathrm{d}\mathbf{x} = C_\mathrm{t} \cdot \int_l \rho(\mathbf{x})\,\mathrm{d}\mathbf{x}$.

We want the optical thickness along the quasi-heterogeneous medium (right side of Eq. 2) to match the thickness along a scaled homogeneous medium (left side), hence we get:

$$\begin{aligned} \sigma_\mathrm{t} \cdot D' &= C_\mathrm{t} \cdot \int_l \rho(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ \sigma_\mathrm{t} \cdot D' &= C_\mathrm{t} \cdot \rho' \cdot \int_l \rho''(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ D' &= \int_l \rho''(\mathbf{x})\,\mathrm{d}\mathbf{x}. \end{aligned} \qquad (2)$$

Eq. 2 enables us to approximate a quasi-heterogeneous medium by a homogeneous one. The equation holds true in case the quasi-heterogeneous medium is in fact homogeneous. In that case, the density ratio would be $\rho''(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathbb{R}^3$ (we compare the exactly same media). Solving Eq. 2 would give us: $D' = \int_l \rho''(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_l 1\,\mathrm{d}\mathbf{x} = D$, which obviously holds true as the length remains the same.

Spatial Spreading We found out that simply assuming $\rho'' \approx \rho$, i.e., $\rho' = 1$, leads to very good visual results. It essentially corresponds to fixing a reference homogeneous medium and only setting the ratio, which is what we want. The Gaussian spatial-spreading deviation W can now be found using Eq. 1 by using the scaled distances:

$$W(D') = W\left(\int_l \rho''(\mathbf{x})\,\mathrm{d}\mathbf{x}\right) \approx W\left(\int_l \rho(\mathbf{x})\,\mathrm{d}\mathbf{x}\right). \qquad (3)$$

3.3 Solving Density Integrals Analytically

General Approach The integral in Eq. 3 corresponds to integrating a density along a path *from a camera to a pixel*, i.e., along the *camera ray*. Solving such integrals analytically is shown by Quílez [10, 11], who derived closed-form solutions for simple exponential and spherical media.

Consider Figure 3: we express the ray $\mathbf{r}$ between the camera origin $\mathbf{c} \in \mathbb{R}^3$ and the pixel position $\mathbf{x} \in \mathbb{R}^3$ *parametrically*. For a parameter t and direction ω, we get: $\mathbf{r}(t) = \mathbf{c} + t \cdot \omega$. In our case, $t \in [0, \|\mathbf{x} - \mathbf{c}\|] = [0, D]$,

where D is the true distance between the camera and the pixel. This approach enables us to integrate the density for the parameter t, as shown in the example below (Eq. 4).

Example: Parametrized Exponential Medium Unlike Quílez, in this paper, we derive a closed-form solution for a more general *arbitrarily rotated* and *translated* exponential medium. By the *exponential medium* we understand a medium where the particle density can be modeled by an exponential function, which is closely related to real media according to the *barometric formula* [1].

As in Figure 3, we suppose that a fixed density ρ' exponentially decreases with a parameter $b > 0$ in a normalized direction $\mathbf{n} \in \mathbb{R}^3$ and is offset by $\mathbf{o} \in \mathbb{R}^3$. Hence the spatially-varying $\rho(\mathbf{x})$:

$$\rho(\mathbf{x}) = \rho' \cdot \exp\left(-b \cdot \langle \mathbf{x} - \mathbf{o}, \mathbf{n} \rangle\right),$$

where $\langle \cdot, \cdot \rangle$ denotes an inner product. We integrate the density $\rho(\mathbf{x}) = \rho(\mathbf{r}(t))$ along the camera ray $\mathbf{r}(t)$ as explained:

$$\begin{aligned} P &= \int_0^D \rho\left(\mathbf{r}(t)\right) \mathrm{d}t = \int_0^D \rho\left(\mathbf{c} + t \cdot \omega\right) \mathrm{d}t \\ &= \rho' \int_0^D \exp\left(-b \cdot \langle \mathbf{c} + t \cdot \omega - \mathbf{o}, \mathbf{n} \rangle\right) \mathrm{d}t \\ &= \rho' \int_0^D \exp\left(-b \cdot \left(\langle \mathbf{c}, \mathbf{n} \rangle + t \langle \omega, \mathbf{n} \rangle - \langle \mathbf{o}, \mathbf{n} \rangle\right)\right) \mathrm{d}t, \end{aligned} \quad (4)$$

where the last equality uses the bilinearity of the inner product. The terms $\langle \mathbf{c}, \mathbf{n} \rangle$, $\langle \omega, \mathbf{n} \rangle$, and $\langle \mathbf{o}, \mathbf{n} \rangle$ are constant for a fixed ray, so the definite integral can be expressed as:

$$P = \rho' \cdot \left[-\frac{\exp\left(-b \cdot \left(\langle \mathbf{c}, \mathbf{n} \rangle + t \langle \omega, \mathbf{n} \rangle - \langle \mathbf{o}, \mathbf{n} \rangle\right)\right)}{b \cdot \langle \omega, \mathbf{n} \rangle} \right]_0^D.$$

In case $\langle \omega, \mathbf{n} \rangle = 0$, when we cannot divide, we simply solve Eq. 4 directly with $\langle \omega, \mathbf{n} \rangle = 0$. Otherwise, we get:

$$P = \rho' \cdot \frac{\exp\left(-b \cdot \langle \mathbf{c} - \mathbf{o}, \mathbf{n} \rangle\right) \cdot \left(1 - \exp\left(-b \cdot D \cdot \langle \omega, \mathbf{n} \rangle\right)\right)}{b \cdot \langle \omega, \mathbf{n} \rangle}.$$

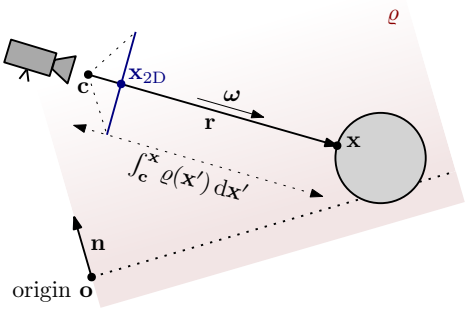

Figure 3: A general exponential medium. The density decreases exponentially along the direction $\mathbf{n}$. A projected pixel $\mathbf{x}_{2D}$ corresponds to the 3D position $\mathbf{x}$. We integrate the density along the camera ray $\mathbf{r}$.

4 Our Filtering Technique

Our screen-space algorithm is based on the technique presented by Elek et al. [3], but we generalize their approach for inhomogeneous media and propose new steps for eliminating artifacts. On the input, we expect an HDR RGB color buffer $\mathsf{L}\colon \mathbb{N}^2 \to \mathbb{R}^3$ and its corresponding distance buffer $\mathsf{D}\colon \mathbb{N}^2 \to \mathbb{R}$, i.e., the world-space distances from the camera. The images are then preprocessed, hierarchically filtered, and finally we compose a single attenuated and scattered image $\mathsf{L}'\colon \mathbb{N}^2 \to \mathbb{R}^3$. For an overview of our whole approach, see the full diagram in Figure 4.

4.1 Preprocessing

Integrated Density and Spread Space From Section 3, we suppose that the densities can be integrated analytically. A *density buffer* $\mathsf{P}\colon \mathbb{N}^2 \to \mathbb{R}$ is constructed, representing the integrated densities from the camera towards the pixel 3D position. We also build an additional *spread-space buffer* $\mathsf{W}\colon \mathbb{N}^2 \to \mathbb{R}$ according to Eq. 3, which corresponds to the spreading widths of the pixels.

$$\mathsf{P}(\mathbf{x}) = \int_{\text{camera}}^{\text{pixel } \mathbf{x}} \rho(\mathbf{x}')\, \mathrm{d}\mathbf{x}', \qquad \mathsf{W}(\mathbf{x}) = W(\mathsf{P}(\mathbf{x})).$$

Attenuated and Scattered Radiance Beer-Lambert law (Section 2) defines how the input radiance L is attenuated w.r.t. the optical thickness $\tau(l)$, which we get as follows:

$$\tau(l) = \int_l \sigma_\mathrm{t}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_l (C_\mathrm{a} + C_\mathrm{s}) \rho(\mathbf{x})\, \mathrm{d}\mathbf{x} = C_\mathrm{t} \int_l \rho(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

By combining this equation with transmittance, we compute the following two buffers. The attenuated image $\mathsf{L}_\mathrm{at}\colon \mathbb{N}^2 \to \mathbb{R}^3$ corresponds to the radiance that was not absorbed. The scattered image $\mathsf{L}_\mathrm{sc}\colon \mathbb{N}^2 \to \mathbb{R}^3$ corresponds to the radiance that was scattered but not absorbed.

$$\begin{aligned} \mathsf{L}_\mathrm{at} &= \exp\left(-C_\mathrm{t} \cdot \mathsf{P}\right) \cdot \mathsf{L}, \\ \mathsf{L}_\mathrm{sc} &= \exp\left(-C_\mathrm{a} \cdot \mathsf{P}\right) \cdot \left(1 - \exp\left(-C_\mathrm{s} \cdot \mathsf{P}\right)\right) \cdot \mathsf{L}. \end{aligned}$$

4.2 Hierarchical Filtering

We want to scatter (spatially spread) the L_sc pixels. A single pixel $\mathbf{x} \in \mathbb{N}^2$ corresponds to radiance $\mathsf{L}_\mathrm{sc}(\mathbf{x})$. Assuming the radiance spreads spatially according to Section 3, the pixel's value is distributed onto its neighbors by a Gaussian distribution centered in $\mathbf{x}$ with a standard deviation $\mathsf{W}(\mathbf{x})$. As different pixels correspond to different deviations W, the Gaussian kernels are *spatially varying*. Unfortunately, filtering with spatially varying kernels is not possible with a fast two-dimensional convolution [3].

Naïve Non-Hierarchical Approaches Two similar approaches can be used for spatially-varying filtering: *splatting* and *gathering* [3, 12]. The splatting approach iterates over all pixels and adds their distribution to their neighbors, requiring multiple writes into a single pixel, which is

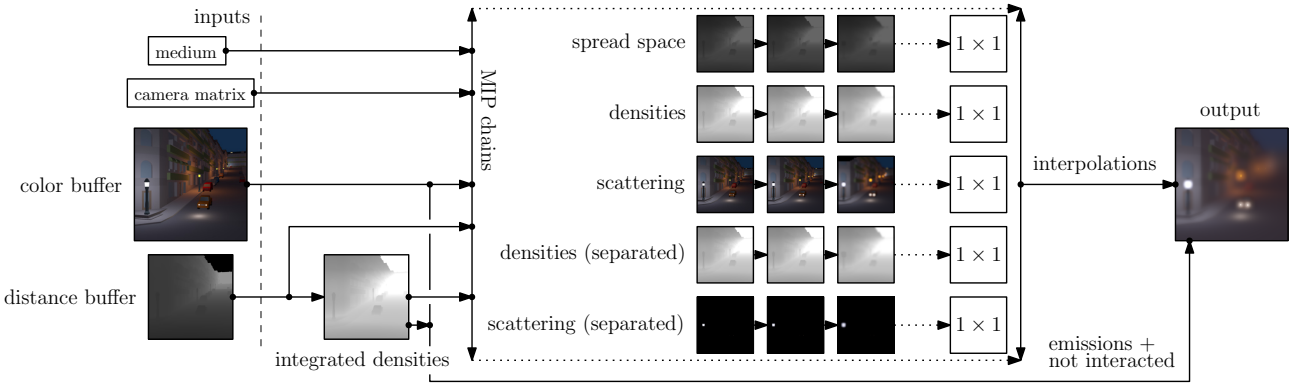

Figure 4: Our filtering technique expects an HDR input buffer and a distance buffer. We then integrate the densities according to the distances. The filtering is hierarchical and several MIP chains are built throughout the process. Finally, we interpolate the data from the MIP chains and get a final output.

not optimal for GPU shaders. The gathering is less effective, but can be implemented in GPU shaders, as for every pixel, it gathers distributions from their neighbors. We implemented a gathering filter for our method as a filtering reference. Processing of a single 1920×1080 frame took more than a minute, unsuitable for real-time rendering.

Hierarchical Gaussian Filtering Fortunately, spatially-varying Gaussian filtering can be approximated using a fast *hierarchical* approach with *MIP maps* [5, 3]: the original image $\mathsf{L}_{\mathrm{sc}} = \mathsf{L}_{\mathrm{sc}}^{[0]}$ corresponds to level 0, the subsequent levels are a power of two smaller than the previous, the last level being K representing a 1×1 image $\mathsf{L}_{\mathrm{sc}}^{[K]}$.

The key idea in the hierarchical filtering is to convolve the subsequent levels using the previous and the same Gaussian kernel. The levels $k \in \{1, \ldots, K\}$ are:

$$\mathsf{L}_{\mathrm{sc}}^{[k]} = \mathsf{L}_{\mathrm{sc}}^{[k-1]} * \mathsf{G},$$

where G is a discrete normalized Gaussian kernel, e.g., using weights $\{0.13, 0.37, 0.37, 0.13\}$ for a 4×4 filter [3].

The width of the filter doubles with every level in the hierarchy [5, 3]. The relation between the deviation W and the level k can be approximated as $W \approx c \cdot 2^{k-1}$, where c is a *scaling constant* (set to 0.86 in our implementation). As the standard deviation increases exponentially, the level itself has to increase logarithmically. Elek et al. [3] propose to approximate the MIP level k as:

$$k(\mathsf{P}) \approx \mathrm{clamp}\left(\log_2 \frac{W(\mathsf{P})}{c}, 0, K\right) \in [0, K].$$

The final filtered scattered image is $\mathsf{L}'_{\mathrm{sc}} = \mathsf{L}_{\mathrm{sc}}^{[k(\mathsf{P})]}$ (composed pixel-by-pixel). The level can be decimal: *one-dimensional interpolation* is used for a level and *two-dimensional interpolation* for a pixel. Both benefit from a fast interpolated texture access on a GPU hardware.

4.3 Eliminating Filtering Artifacts

Luminance Weighting Imagine a very bright object with a low standard deviation W, e.g., the yellow lantern in Figure 5b. When there is a dark object in the background and its standard deviation W is much higher, the luminance of the bright pixels *leaks* into their neighborhood.

When building the MIP chain, the luminance of the bright object is propagated into high MIP levels, even though the standard deviation of the object is small. When fetching the colors for the dark background, the bright luminance incorrectly leaks into the dark area. This problem is called *illumination leaking* [3] and can be seen in Figure 5bc as an incorrectly huge yellow blur of the lantern.

Masking idea Suppressing the illumination leaking is explained by Elek et al. [3]. When building the MIP chain, we introduce a new auxiliary mask M and build the MIP chain in a modified way:

$$\mathsf{M}^{[k]} = \mathrm{smoothstep}\left(T, (1+\varepsilon) \cdot T, \mathsf{W}^{[k-1]}\right),$$
$$\mathsf{L}_{\mathrm{sc}}^{[k]} = \left(\mathsf{M}^{[k]} \cdot \mathsf{L}_{\mathrm{sc}}^{[k-1]}\right) * \mathsf{G}.$$

The masking threshold distance T and width ε control the masking, where $T = c \cdot 2^{k-1}$ and $\varepsilon \geq 0$ should be set according to the scene ($\varepsilon = 2$ in our implementation).

Spread-space MIP chain The spread-space $\mathsf{W}^{[0\ldots K]}$ should correspond to the standard deviation W averaged for neighboring pixels. Elek et al. [3] weight the pixels with their luminance y, where U is a uniform distribution:

$$\mathsf{Y}^{[k]} = y(\mathsf{L}_{\mathrm{sc}}^{[k-1]}), \qquad \mathsf{W}^{[k]} = \frac{\left(\mathsf{Y}^{[k]} \cdot \mathsf{W}^{[k-1]}\right) * \mathsf{U}}{\mathsf{Y}^{[k]} * \mathsf{U}}.$$

Blurring The filtered $\mathsf{L}_{\mathrm{sc}}^{[k(\mathsf{W})]}$ may suffer from noticeable discontinuities. They can be suppressed by applying the Gaussian MIP map blurring to our densities P. We build

another MIP chain $\mathsf{P}^{[0\ldots K]}$ and then fetch as proposed by Elek et al. [3]:

$$\mathsf{P}^{[k]} = \mathsf{P}^{[k-1]} * \mathsf{G}, \qquad k' = k\left(\mathsf{P}^{[k(\mathsf{P})]}\right), \qquad \mathsf{L}'_{\mathrm{sc}} = \mathsf{L}_{\mathrm{sc}}^{[k']}.$$

Optional Proposal: Radiance Separation In HDR images, certain pixels may be much brighter than the rest. When the density $p_1 \in \mathsf{P}$ of a bright foreground is much lower than the density $p_2 \in \mathsf{P}$ of a neighboring dark background, the blurring step smoothes this large discontinuity. This can completely eliminate the foreground from high levels of $\mathsf{P}^{[0\ldots K]}$. It essentially means that the dark area of our image remains dark and the blur of a bright foreground is "cut" at the edge (Figure 5d).

To heuristically *separate* bright pixels from the original input L, we split the input L into two different scattering images: L_{sc} denotes the scattering image without the separated pixels and $\mathsf{L}_{\mathrm{ssc}}$ with the separated pixels:

$$\mathsf{L}_{\mathrm{ssc}} = \mathrm{e}^{-C_{\mathrm{a}}\cdot\mathsf{P}} \cdot \left(1 - \mathrm{e}^{-C_{\mathrm{s}}\cdot\mathsf{P}}\right) \cdot \mathsf{M} \cdot \mathsf{L},$$
$$\mathsf{L}_{\mathrm{sc}} = \underbrace{\mathrm{e}^{-C_{\mathrm{a}}\cdot\mathsf{P}} \cdot \left(1 - \mathrm{e}^{-C_{\mathrm{s}}\cdot\mathsf{P}}\right)}_{\text{original weight of } \mathsf{L}_{\mathrm{sc}}} \cdot (1 - \mathsf{M}) \cdot \mathsf{L},$$

$$\mathsf{M} = \underbrace{\left(\text{smoothstep}\left(T_y, T_y + \varepsilon_y, \mathsf{Y}^{[0]}\right)\right)}_{\text{high luminance}} \cdot \underbrace{\left(1 - \text{smoothstep}\left(T_{d'} - \varepsilon_{d'}, T_{d'}, \mathsf{P}^{[0]}\right)\right)}_{\text{low integrated density}},$$

where M is based on smoothly selecting the pixels with *high* absolute RGB luminance $y \in \mathsf{Y}^{[0]}$ and *low* integrated density $p \in \mathsf{P}^{[0]}$ and the parameters are thresholds.

The image L_{sc} is filtered using the same luminance weighting technique described earlier. However, we build the MIP chain $\mathsf{L}_{\mathrm{ssc}}^{[0\ldots K]}$ differently as the image is mostly black and only contains a few separated pixels. To prioritize the densities of the bright separated pixels, we construct $\mathsf{P}_{\mathrm{ssc}}^{[0\ldots K]}$ based on luminance weighted averaging:

$$\mathsf{L}_{\mathrm{ssc}}^{[k]} = \mathsf{L}_{\mathrm{ssc}}^{[k-1]} * \mathsf{G}, \qquad \mathsf{Y}_{\mathrm{ssc}}^{[0\ldots K]} = y(\mathsf{L}_{\mathrm{ssc}}^{[0\ldots K]}),$$
$$\mathsf{P}_{\mathrm{ssc}}^{[k]} = \frac{\left(\mathsf{Y}_{\mathrm{ssc}}^{[k]} \cdot \mathsf{P}_{\mathrm{ssc}}^{[k-1]}\right) * \mathsf{U}}{\mathsf{Y}_{\mathrm{ssc}}^{[k]} * \mathsf{U}}.$$

Result The process to building the final $\mathsf{L}'_{\mathrm{ssc}}$ is similar to $\mathsf{L}'_{\mathrm{sc}}$, except we need to take special care of choosing the correct level. This heavily depends on how many pixels were actually separated in our scene. We introduce an arbitrary parameter k_{ssc} ($0 \le k_{\mathrm{ssc}} \le K$) set to $k_{\mathrm{ssc}} = 0.7 \cdot K$ in our implementation. The fetching equation is:

$$k' = \ell\left(\mathsf{P}_{\mathrm{ssc}}^{[k_{\mathrm{ssc}}]}\right), \qquad \mathsf{L}'_{\mathrm{ssc}} = \mathsf{L}_{\mathrm{ssc}}^{[k']}.$$

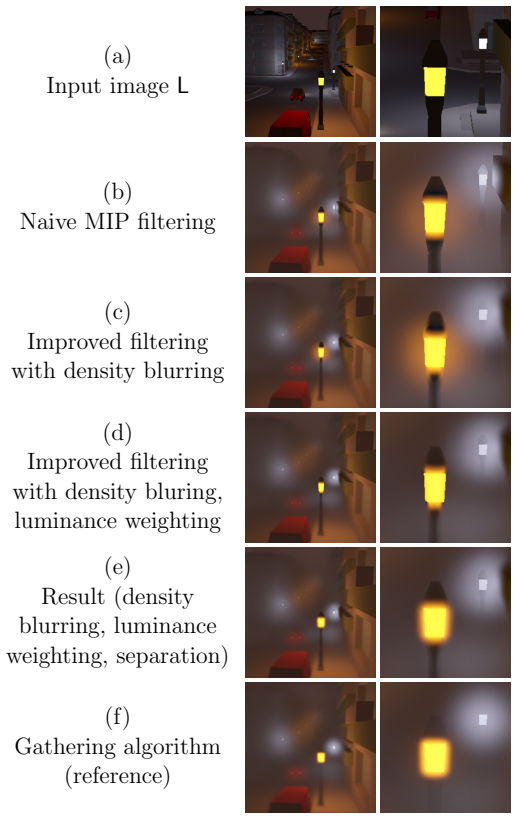

Figure 5: Comparison of filtering the input image (a) with different approaches (b)-(e), where (e) is the final result. The gathering algorithm (f) can be understood as a filtering reference but is too slow for real-time use.

4.4 Final Compositing

The final step is to compose the output L'. It is based on three images that we have computed during the previous steps: L_{at} corresponds to the radiance that reaches the camera without any interactions, $\mathsf{L}'_{\mathrm{sc}}$ and $\mathsf{L}'_{\mathrm{ssc}}$ together represent the radiance scattered on the way to the camera. The final output corresponds to adding the results together:

$$\mathsf{L}' = \mathsf{L}_{\mathrm{at}} + \mathsf{L}'_{\mathrm{sc}} + \mathsf{L}'_{\mathrm{ssc}}.$$

5 Implementation

We implemented the filtering in GPU shaders in our custom engine based on bgfx[1], where we also measured the performance. To prove that the method itself can be added to existing frameworks and engines, we have also successfully implemented our post-processing filtering in Unity[2] and used it in existing scenes without any difficulties.

In our implementation, floating-point GPU buffers are used as we need an HDR input and floating-point precision intermediate results. It is beneficial to use RGBA buffers, where the alpha channel can store the integrated densities.

[1] https://github.com/bkaradzic/bgfx
[2] https://unity3d.com/

Resolution	Filter size	Interpolation technique			Separation (optional)
		Linear-bilinear	Linear-bicubic	Cubic-bicubic	Linear-bicubic
1280×720	2×2	0.9 ms	1.1 ms	1.6 ms	+0.6 ms
	4×4	1.9 ms	♡ 2.1 ms	2.6 ms	+1.0 ms
	6×6	3.7 ms	3.9 ms	4.3 ms	+2.3 ms
1920×1080	2×2	2.0 ms	2.5 ms	3.4 ms	+1.2 ms
	4×4	4.3 ms	♡ 4.8 ms	5.6 ms	+2.1 ms
	6×6	8.3 ms	8.7 ms	9.7 ms	+3.6 ms

Table 1: Total times for post-processing a single frame using our method on NVIDIA GeForce GTX 660. In practice, the 4×4 linear-bicubic filtering (♡) offers the best compromise between price and quality. The times of the optional separation step (last column, see Section 4.3) have to be added to the total if the step is enabled.

In animated scenes, it is critical to use a smooth interpolation method for fetching from the MIP maps, i.e., at least a linear-bicubic interpolation. In theory, we need 16 texture reads for a bicubic interpolation, but we accelerated the interpolation to 4 reads benefiting from the hardware bilinear filtering as shown by Sigg and Hadwiger [13]. Furthermore, not computing all MIP levels down to 1×1, but stopping sooner, is acceptable in a lot of smaller scenes and can improve the performance.

Performance Let us note that our demo application is a prototype and the fragment shaders are not primarily optimized for performance. In production, lot of branching and parameters could be removed and/or optimized.

The total times to process a single input image and render the participating effects using our method are detailed in Table 1. The times were measured using scenes with a lot of point lights and emissive textures as can be seen on the screenshots throughout the paper (Figures 1, 6).

Rendering the participating effects takes only a few milliseconds even for the Full HD resolution using a rather old GPU. The processing times of 2.1 ms and 4.8 ms (♡) for HD and Full HD resolutions respectively correspond to the performance measured by Elek et al. [3] for homogeneous media. The framerate remained stable and the application was fully responsive and usable in real time.

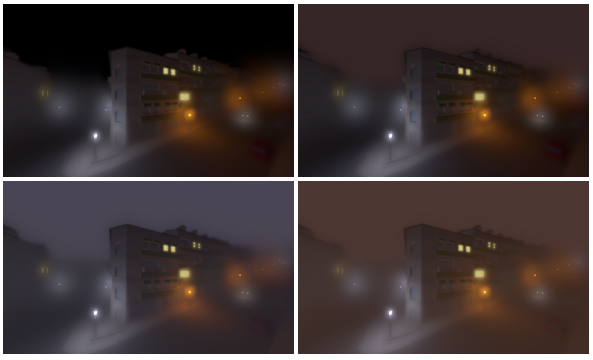

Figure 6: The same scene rendered using our method. The ambient color in the 4 pictures was influenced by changing the sky color and mixing the absorption with emissions.

6 Discussion

6.1 Comparison to Existing Solutions

There exist various **empirical approaches** to rendering participating media. Commonly, simple *alpha blending* with a constant fog color is used [10, 14]. It is not physically based, cannot blur the light sources and the geometry and does not conserve energy. More advanced approaches use *billboards* and *particle effects* [8], but they are only useful for local effects such as smoke and have the same disadvantages as noted above.

Our solution is built on a *physically-based* post-processing method of **Elek et al.** [3] capable of rendering multiple-scattering effects in real time, but only in homogeneous media. In this paper, we extended their method for inhomogeneous media and by adding additional steps for eliminating the filtering artifacts with HDR inputs.

Shinya et al. [12] recently (2016) presented a novel physically-based post-processing solution built on a different background. They also approximate *multiple scattering* in *inhomogeneous media*. They even claim to achieve more precise results than Elek et al. when compared to a path-tracing reference. Unfortunately, according to their article, rendering a single 640×480 frame on a comparable hardware takes 100-110 ms, which is approximately two orders of magnitude slower than our approach.

Wronski [14] presented a solution successfully used in AAA video games, e.g., in *Assasin's Creed IV*. Their post-processing *volumetric fog* runs smoothly on game consoles such as Xbox One. They represent the camera frustum in a low-resolution 3D texture. Unlike our method, their solution accumulates single-in-scattering from light sources directly, so it does not blur the geometry and does not handle emissive materials. However, it supports volumetric shadows and arbitrary heterogeneous media as they use ray marching to handle the densities.

Finally, we mention a completely different approach of **Mitchell** [6] used for fast screen-space rendering of volumetric shadows in homogeneous media. Unfortunately, it only supports a single light source (e.g., sun) and is not intended for multiple scattering in the whole scene.

6.2 Limitations and Future Work

Temporal Coherence Our method provides good temporal coherence in animated scenes, which can be verified in a supplemental video. However, it still suffers from a common limitation of screen-space approaches: it can only work with information present in the inputs.

When certain objects get occluded or move out of the image, they may pop and disappear in subsequent frames causing noticeable flickering. This occurs especially with distant light sources in sub-pixel regions. Approaches such as *deep screen space* by Nalbach et al. [7] could be implemented and evaluated to overcome these problems.

Participating Effects Before Reflections We only process the radiance that has already been reflected from surfaces, i.e., on the path between surfaces and the camera. However, in real media, the attenuation and scattering also happen on the paths between light sources and surfaces. For a future work, one could try implementing the participating effects in the light shaders as well.

Ambient Term The overall atmosphere of the scenes may be influenced by the total environmental illumination caused by all light sources together. In Figure 6, we simulate the ambient color by adding an emissive term to the medium. For a future work, the ambient term could be calculated dynamically from the scattering buffers.

Volumetric Occlusions Similarly to the methods of Elek et al. [3] and Shinya et al. [12], we ignore volumetric occlusions, hence cannot render phenomena such as crepuscular rays. For a future work, additional steps could be introduced to detect volumetric collisions in screen-space, e.g., similarly to Mitchell [6].

Media Simplifications We only consider media with analytically integrable and smooth density functions. The filtering, however, accepts any integrated densities. For a future work, one could try animating the media using random noise. Furthermore, a different approximation based on angular spreading could provide better results.

6.3 Conclusion

We have presented a novel physically-based approach for real-time rendering of participating media. Unlike the existing solutions, ours offers fast multiple scattering in inhomogeneous media. We have verified that our approach can be implemented in existing engines and enables us to render visually convincing animations in real time.

Acknowledgment

I would like to thank Oskar Elek for all the valuable advice he offered during the creation of this work. I also thank the Kenney Group, Jim van Hazendonk, and Christoph Peters for releasing their 3D assets to the Public Domain, which allowed us to use them in our application and screenshots.

This project has received funding from the European Union's Horizon 2020 research and innovation programme, under the Marie Skłodowska-Curie grant agreement No 642841 (DISTRO), and the Czech Science Foundation grant 16-18964S.

References

[1] M. N. Berberan-Santos, E. N. Bodunov, and L. Pogliani. On the barometric formula. *American Journal of Physics*, 65(5):404–412, 1997.

[2] O. Elek. *Efficient Methods for Physically-Based Rendering of Participating Media*. PhD thesis, Max Planck Institute for Computer Science, 2016.

[3] O. Elek, T. Ritschel, and H.-S. Seidel. Real-time screen-space scattering in homogeneous environments. *IEEE CG&A*, 2013.

[4] W. Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. San Diego, 2008.

[5] S. Lee, G. J. Kim, and S. Choi. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):453–464, 2009.

[6] K. Mitchell. Volumetric light scattering as a post-process. *GPU Gems 3*, pages 275–284, 2007.

[7] O. Nalbach, T. Ritschel, and H.-P. Seidel. Deep screen space. *Proc. I3D*, 2014.

[8] T. Persson. Practical particle lighting, 2012.

[9] S. Premože, M. Ashikhmin, R. Ramamoorthi, and S. Nayar. Practical rendering of multiple scattering effects in participating media. *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 363–374, 2004.

[10] Í. Quílez. Better fog. `http://www.iquilezles.org/www/articles/fog/fog.htm`, 2010. Accessed: 2018-02-08.

[11] Í. Quílez. Sphere density. `http://www.iquilezles.org/www/articles/spheredensity/spheredensity.htm`, 2015. Accessed: 2018-02-08.

[12] M. Shinya, Y. Dobashi, M. Shiraishi, M. Kawashima, and T. Nishita. Multiple scattering approximation in heterogeneous media by narrow beam distributions. *Computer Graphics Forum*, 35(7):373–382, 2016.

[13] C. Sigg and M. Hadwiger. Fast third-order texture filtering. *GPU Gems 2*, pages 313–329, 2005.

[14] B. Wronski. Volumetric fog: Unified compute shader-based solution to atmospheric scattering, 2014.