UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PPGC

IGOR BATISTA FERNANDES

An approximation to multiple scattering in volumetric illumination towards real-time rendering

Master's Dissertation

Advisor: Prof. Dr. Marcelo Walter
Coadvisor: Prof. Dr. Pierre Poulin

Porto Alegre
November 2021

ACKNOWLEDGEMENTS

I would like to start by thanking my parents and sister, without their foundation and help I would not be who I am today. I am also grateful for the support and help from my friends throughout the entirety of our time passed together, the ones I have met since birth and the recent ones, you are all very dear to me. I want to also thank Marcelo Walter for welcoming me in Porto Alegre and giving valuable advice and motivation during the whole journey. Likewise I would like to thank Pierre Poulin for welcoming me in Montreal and also giving valuable advice and support throughout my brief stay in Canada. This work would not have been what it is if all of you were not a part of it. Furthermore I need to thank the UFRGS, which gave all the structure and aide when necessary, Cnpq, for the scholarship that helped sustain myself in Porto Alegre, ELAP and Globalink for their scholarships that made my stay in Montreal possible, enhancing not only the quality of my work, but myself as a human being.

ABSTRACT

Many volumetric illumination techniques for volume rendering were developed throughout the years. However, there are still many constraints regarding the computation of multiple scattering path tracing in real-time applications due to its natural complexity and scale. Path tracing with multiple scattering support can produce physically correct results but suffers from noise and low convergence rates. This work proposes a new real-time algorithm to approximate multiple scattering, usually only available in offline rendering production, to real-time. Our approach explores the human perceptual system to speed up computation. Given two images, we use a CIE metric stating that the two will be perceived as similar to the human eye if the Euclidean distance between the two images in CIELAB color space is smaller than 2.3. Hence, we use this premise to guide our investigations when changing ray and bounce parameters in our renderer. Our results show that we can reduce from 10^5 to 10^4 Samples Per Pixel (SPP) with a negligible perceptual difference between both results, allowing us to cut rendering times by 10 whenever we divide SPP by 10. Similarly, we can reduce the number of bounces from 1000 to 100 with a negligible perceptual difference while reducing rendering times by almost half. We also propose a new algorithm in real-time, Lobe Estimator, that approximates these behaviors and parameters while performing twice as faster as the classic Ray Marching technique.

Keywords: Computer graphics. Volumetric rendering. Volumetric illumination. Path tracing. Light transport. Monte Carlo path tracing. Global illumination. Real-time.

RESUMO

Muitas técnicas de ilmuninação volumétrica foram desenvolvidas ao longo dos anos. Entretanto, ainda há muitas restrições na computação de *multiple scattering* em aplicações de tempo real usando *path tracing*, devido à sua complexidade e escala. *Path tracing* com suporte a *multiple scattering* é capaz de produzir resultados fisicamente corretos, mas sofre de ruídos e baixa taixa de convergência. Portanto, este trabalho propõe um novo algoritmo de tempo real para aproximar *multiple scattering*, usado em *offline rendering*. Nossa abordagem irá explorar o sistema perceptual visual humano para acelerar a computação. A partir de duas imagens, nós usamos a métrica da CIE que afirma que duas imagens são percebidas como similar ao olho humano se a distância Euclidiana das duas imagens no espaço de cores CIELAB for menor que 2.3. Dessa forma, nós usamos essa premissa para guiar nossas investigações quando alterando os parâmetros de *Samples Per Pixel* (SPP) e *bounces* nos renderizadores. Nossos resultados mostram que podemos reduzir de 10^5 para 10^4 *Samples Per Pixel* (SPP) com uma diferença perceptual negligenciável entre ambos paramêtros, permitindo reduzir o tempo de renderização por 10 a cada vez que dividimos o SPP por 10. Similarmente, nós podemos reduzir o número de *bounces* de 1000 para 100 com uma diferença perceptual negligenciável, enquanto reduzindo o tempo de renderização por quase metade. Nós também propusemos um novo algoritmo em tempo real, Lobe Estimator, que permite aproximar esses comportamentos e paramêtros enquanto permformando duas vezes mais rápido que o clássico Ray Marching.

Palavras-chave: Computer Graphics, Volumetric rendering, Volumetric illumination, Path tracing, Light transport, Monte Carlo path tracing, Global illumination.

LIST OF ABBREVIATIONS AND ACRONYMS

sr	Steradian, also known as square radian. Unit of solid angle.
RTE	Radiative Transfer Equation.
VRE	Volume Rendering Equation.
HG	Henyey-Greenstein phase function.
SPP	Samples Per Pixel.
BSDF	Bidirectional Scattering Distribution Function.
PDF	Probability Density Function.
CDF	Cumulative Distribution Function.
MIS	Multiple Importance Sampling.
JND	Just Noticeable Difference.
AABB	Axis Aligned Bounding Box.

GLOSSARY

Radiance — Radiant flux per unit solid angle and unit area projected in a direction. Measured as Watts per steradian per square meter ($\frac{W}{sr \cdot m^2}$) (DORSEY; RUSHMEIER; SILLION, 2008).

Irradiance — Radiant flux per unit time and area arriving at a surface. Measured as Watts per square metre ($\frac{W}{m^2}$) (DORSEY; RUSHMEIER; SILLION, 2008).

Optical depth — Also known as optical thickness, is the natural logarithm of the ratio of incident to transmitted radiant power through a material. It is dimensionless and written as $\tau = ln(\frac{incident\ radiance}{transmitted\ radiance}) = -ln(T)$, where T is the material's transmittance.

Voxel — A unit of graphic information that defines an element of a regular grid in three-dimensional space. The word is derivated from the combination of *vo* (volume) and *xel* (pixel).

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Volumetric rendering is a field of computer graphics focused on techniques to render participating media, i.e., large spatial volumes filled with sparse particles, e.g., dust, water droplets, clouds, and smoke (FONG et al., 2017). It is also a process of calculating how light interacts with these media and what physical phenomena occur due to this interaction. An example of ubiquitous participating media, clouds, are shown in Figures 1.1 and 1.2.

Techniques used to calculate light transport within a volume are based on the assumption that the volume is composed of many tiny and sparse particles that its rendering model is best modeled as a probabilistic process, rather than calculating each interaction with every single particle (PHARR; JAKOB; HUMPHREYS, 2016). By following this strategy, it is necessary to consider three properties while calculating light interactions with a medium: absorption, emission, and scattering.

In the real-time rendering community, the pursue of realistic interactive image synthesis has been one of the major goals since its very beginning (JÖNSSON et al., 2014). In more recent years, there has been an effort to capture the very challenging property of multiple scattering, including anisotropic scattering, to real-time rendering. Too often, it faced substantial limitations due to grid sizes (ELEK et al., 2014). Although it is possible to render the effect of multiple scattering in real-time with constraints, it is still a challenging phenomenon to account for due to its complexity. It is too complex to trace all the necessary rays and compute all the properties of a single participating medium. It turns out even harder when we consider mixed media and other objects in the scene that are interacting with each other.

There are many different volumetric illumination techniques for interactive volume rendering (JÖNSSON et al., 2014). Some of the most important techniques are discussed in Chapter 3. The most common technique for light transport used for both participating media and surfaces used today in production involves path tracing (FONG et al., 2017). It uses Monte Carlo integration to simulate global illumination, which requires a considerable number of rays and bounces to converge to a satisfactory image with low noise. To overcome this problem, the industry has adopted a variety of methods of sampling and denoising algorithms (HOFMANN et al., 2021), but it is still a massive bottleneck for real-time rendering.

This work proposes to bring together volumetric path tracing algorithms and per-

ceptual principles to synthesize images faster. We will use the device-independent CIELAB metric as a tool to estimate the threshold values at which the human eye can no longer perceive differences between the proposed ground truth and the target being tested (MAHY; EYCKEN; OOSTERLINCK, 1994). By real-time, we target less than 16 milliseconds per frame (ELLIS et al., 2004).

This dissertation is organized as follows: Chapter 2 lays the foundations of volumetric rendering theory, followed by the related work in Chapter 3. In Chapter 4, we briefly discuss the most common algorithms used in production volumetric rendering. In Chapter 5, we conduct a visual analysis based on the CIELAB metric, exploring how the numbers of samples per pixel and bounces affect the rendering quality of volumetric scenes. Last, in Chapter 6, we propose a new algorithm, Lobe Estimator, that approximates volumetric rendering algorithms used in offline rendering, bringing them to real-time while also performing better than the classic ray marching used for real-time applications.

Figure 1.1: Clouds, a classic example of participating media. Photo taken at Chute Montmorency in Quebec City, Canada.

Source: The author.

Figure 1.2: Clouds, a classic example of participating media. Photo taken at Serra da Canastra in Minas Gerais, Brasil.

Source: The author.

2 THEORY ON VOLUMETRIC RENDERING

A participating medium can be homogeneous or heterogeneous, i.e., its properties of absorption, emission, and scattering are constant or non-constant throughout the volume (PHARR; JAKOB; HUMPHREYS, 2016). A medium can also be anisotropic or isotropic, i.e., the scattering probability does or does not depend on incoming light direction, while allowing asymmetric scattering, calculated by the phase function.

The notation for variables and functions used in the process of calculating light interactions within a participating medium is shown in Table 2.1.

Table 2.1: All mathematical notations used to calculate light transport through a participating media.

Symbol	**Description**	**Unit**
ξ	Random number, usually in the interval [0, 1)	unitless
σ_a	absorption coefficient	m^{-1}
σ_s	scattering coefficient	m^{-1}
σ_t	extinction coefficient ($\sigma_t = \sigma_a + \sigma_s$)	m^{-1}
ρ	albedo	unitless
$f_p(x, \omega, \omega')$	phase function at point x given incoming direction ω and exiting direction ω'	sr^{-1}
$L(x, \omega)$	radiance at point x in direction ω	$W \cdot m^{-2}$
$L_s(x, \omega)$	radiance scattered at point x in direction ω	
$L_e(x, \omega)$	radiance emitted at point x in direction ω	
$L_b(x, \omega)$	background radiance at point x in direction ω	
$Tr(x, x')$	Transmittance from point x to x'	

2.1 Mathematical Model and Foundation

The mathematical model used to render participating media based on the Radiative Transfer Equation (RTE) (Chandrasekhar, 1950), more precisely on the optical model proposed by (Max, 1995), is called Volume Rendering Equation (VRE) (FONG et al., 2017) when adapted to the rendering field. The VRE can separately incorporate the properties of absorption, emission, and scattering. The original RTE from which we derive the VRE is written as follows:

$$(\omega\cdot\nabla)L(x,\omega) = \underbrace{-\sigma_t(x)L(x,\omega)}_{\text{Absorption and Out-scattering}} + \underbrace{\sigma_a(x)L_e(x,\omega)}_{\text{Emission}} + \underbrace{\sigma_s(x)\int_{S^2} f_p(x,\omega,\omega')L(x,\omega')d\omega'}_{\text{In-scattering}} \tag{1}$$

where the absorption and out-scattering terms are originally written as $(\omega\nabla)L(x,\omega) = -\sigma_a(x)L(x,\omega)$ and $(\omega\nabla)L(x,\omega) = -\sigma_s(x)L(x,\omega)$, respectively. Hence, because they can be combined into the extinction coefficient σ_t, both expressing losses in radiance, we can simplify then as $L(x,\omega) = -\sigma_t(x)L(x,\omega)$. To further simplify we rename the integral part of the in-scattering term to $L_s(x,\omega) = \int_{S^2} f_p(x,\omega,\omega')L(x,\omega')d\omega'$, yielding the final RTE:

$$L(x,\omega) = -\sigma_t(x)L(x,\omega) + \sigma_a(x)L_e(x,\omega) + \sigma_s(x)L_s(x,\omega). \tag{2}$$

Note that the $L_s(x,\omega)$ term integrates over the whole unit sphere S^2. By finding a suitable boundary condition and evaluating the endpoints of the function curve we can integrate both sides of Equation 2, yielding the Volumetric Rendering Equation (VRE) given in Equation 3.

$$L(x,\omega) = L_b(x_b,\omega_b)Tr(x,x') + \int_x^{x'} Tr(x,x')[\sigma_a(x)L_e(x,\omega) + \sigma_s(x)L_s(x,\omega)]dx \tag{3}$$

where $L_b(x_b,\omega)$ is the incident radiance entering the volume and $Tr(x,x')$ is the transmittance function in terms of the extinction coefficient σ_t. The transmittance function describes how radiance decays through the volume, and it is written as:

$$Tr(x,x') = e^{-\int_x^{x'}\sigma_t(x)dx}. \tag{4}$$

Since the nature of VDB datasets used in volumetric rendering are discrete, the integrals are approximated numerically, commonly with a Monte Carlo method. A number of participating media are available under the VDB format at (FOUNDATION,). The four major volumetric properties, i.e., absorption, emission, in-scattering, and out-scattering, are discussed in Sections 2.2, 2.3, 2.4, and 2.5.

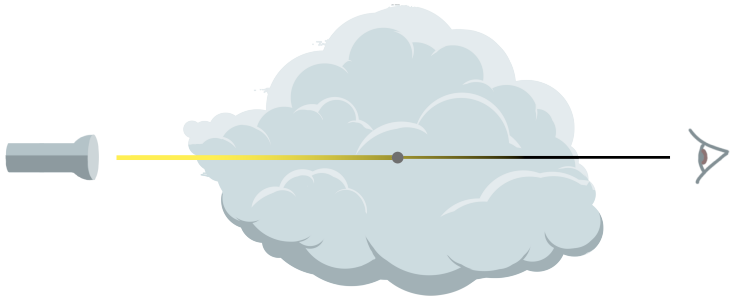
2.2 Absorption

Absorption, denoted by σ_a, occurs when light passes through a particle and is converted to another form of energy, e.g. heat. This effect is mostly noticeable on thicker media such as, for example, dense smoke produced from a fire or an explosion, where light mostly fails to traverse the media and reach the observer due to a high absorption rate in the volume. As described in Table 2.1, the unit of σ_a is the reciprocal distance m^{-1}, which means that it can take any positive value, and thus represents the probability density that light radiance is absorbed per unit distance traveled in the medium. An illustrative example is shown in Figure 2.1 where, as the light beam traverses the cloud, it gradually loses energy due to absorption by the participating media.

To calculate the absorption we use Equation 5, where d is the distance a ray travels in the direction ω, and $x = 0$ is the starting point. This formula comes from Beer's law. When the medium is homogeneous, thus σ_a is constant, we can evaluate the equation simply as a function of the form e^{-x}, shown in Figure 2.2.

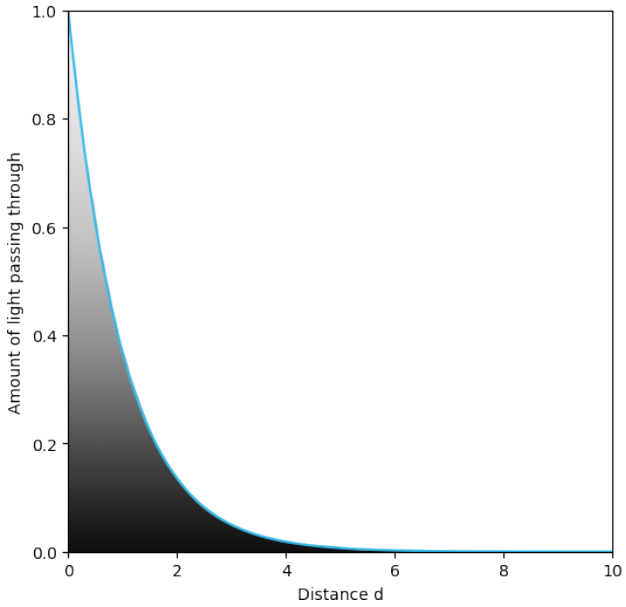
$$e^{-\int_{x=0}^{d} \sigma_a(x+t\omega)dx}. \tag{5}$$

Figure 2.1: Beam of light traversing a cloud and losing energy by absorption when colliding with a particle.



Source: The Author.

Figure 2.2: Absorption as a function of the form e^{-x}, where the y axis is the amount of energy passing through and the x axis is the distance traveled inside the medium. Therefore, it represents the decay of radiance by absorption along traveled distance d.



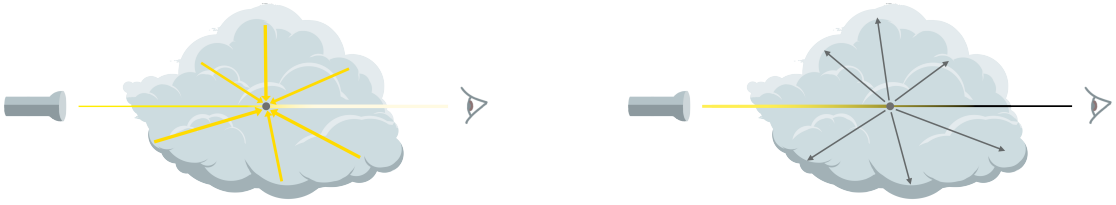
Source: The Author.

2.3 Scattering

Scattering occurs when a light ray bounces inside the medium as a consequence of collisions with particles, as illustrated in Figure 2.3. It can have two effects on the radiance of the light beam, and is classified in two forms:

- Out-scattering: When the light ray hits a particle inside the medium it may bounce off the path, thus reducing radiance.
- In-scattering: When another light ray interacts with the current light beam, thus increasing radiance.

Figure 2.3: Effects of in- and out-scattering as a light ray passes through the medium until reaching the observer.



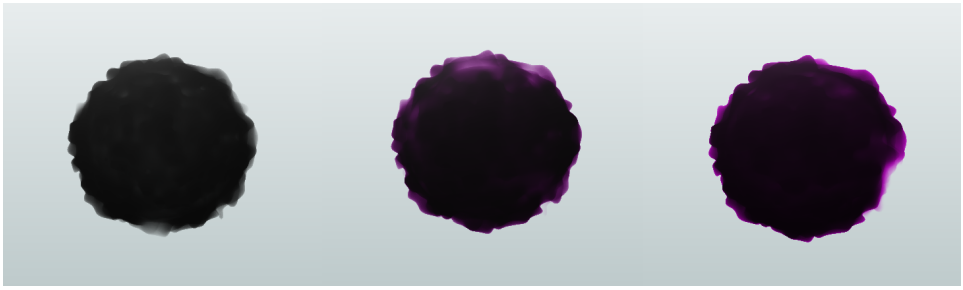
(a) In-scattering.

(b) Out-scattering.

Source: The Author.

The probability of an out-scattering event occurring per unit distance is given by the scattering coefficient σ_s. Since absorption is also an event that reduces radiance we can define the extinction coefficient σ_t as the sum of absorption and out-scattering coefficients: $\sigma_t = \sigma_a + \sigma_s$. An example of these effects is shown in Figure 2.4.

Figure 2.4: An example of a homogeneous participating medium with absorption values σ_a of 0 and scattering σ_s values of $\sigma_s(0.1, 0.1, 0.1)$, $\sigma_s(0.1, 0.3, 0.1)$, and $\sigma_s(0.1, 0.9, 0.1)$, respectively.

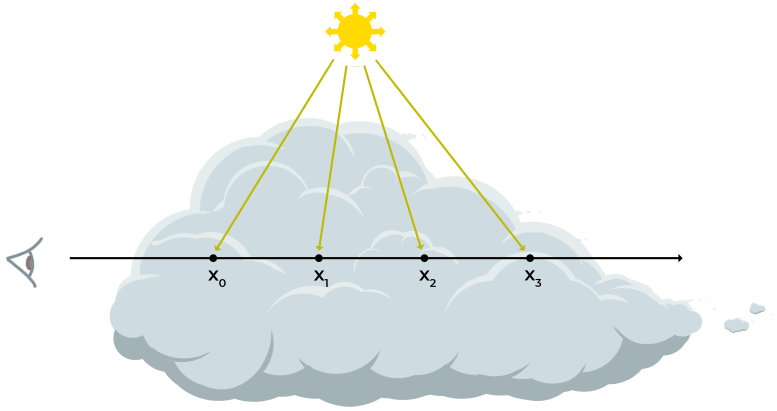


Source: The Author.

2.3.1 Single Scattering

If each sample point along the light beam keeps the direction constant, the method is categorized as handling only single scattering. This process is illustrated in Figure 2.5. Single scattering is commonly used with the ray marching technique, where both the step size and ray direction inside the medium are constant.

Figure 2.5: Single scattering where the beam traverses the medium along a straight line without any disturbance along its path direction.

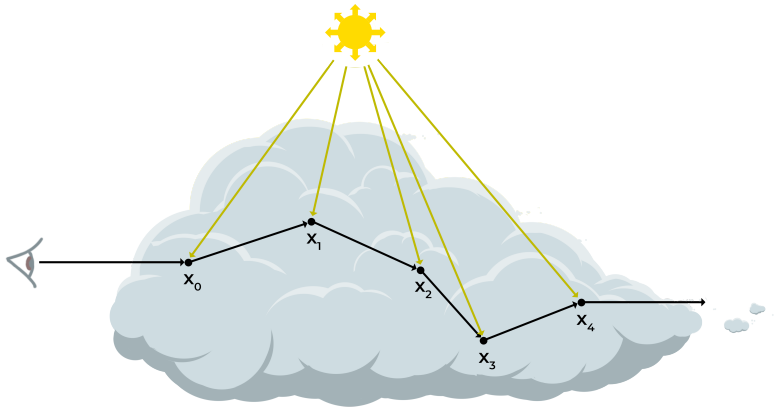


Source: The Author.

2.3.2 Multiple Scattering

If each sample point along the beam may change the ray direction, then the method is categorized as handling multiple scattering. This process is illustrated in Figure 2.6. Multiple scattering is more commonly used in offline renderers, where the constraint of time is more flexible, thus allowing for more processing power to take place.

Figure 2.6: Multiple scattering where the beam traverses the medium with changes in its direction.



Source: The Author.

2.4 Extinction

Extinction, also known as attenuation, is calculated as the sum of absorption and out-scattering ($\sigma_t = \sigma_a + \sigma_s$), where absorption is a function removing photons and out-scattering is a function redirecting photons away from the current path direction.

There are other two important terms related to the extinction coefficient σ_t. The first one is albedo, always between 0 and 1, which describes the probability of a scattering event at each sampling point. An albedo of 0 means that all radiance is absorbed, and an albedo of 1 means that no absorption occurs. Its equation is defined as:

$$\rho = \frac{\sigma_s}{\sigma_t}.$$

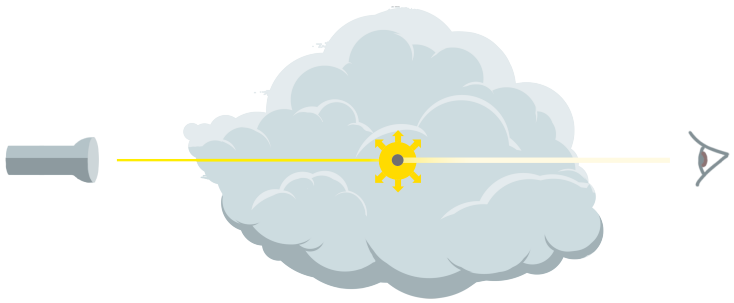
The other term is the mean free path, written as $\frac{1}{\sigma_t}$, which is the average distance that a ray travels in the medium before interacting with any particle. These two variables control the beam transmittance, which gives the fraction of radiance transmitted between two points.

$$e^{-\int_0^d \sigma_t(t)dt}.$$

2.5 Emission

Emission, the opposite of absorption, increases the amount of radiance along a ray that passes through the medium. It increases radiance through a process of thermal or nuclear conversion of energy into light. This process is illustrated in Figure 2.7.

Figure 2.7: Beam of light traversing a cloud and increasing its total energy by emission from colliding with a particle emitting light.



Source: The Author.

2.6 Phase Function

There are several phase functions, each being suitable for a different use case. The most common in rendering are isotropic, Henyey-Greenstein (HENYEY; GREENSTEIN, 1941), commonly abbreviated as HG, and Mie. A phase function describes the angular distribution of rays at a scattering event, which is a point sample along the light path. The effects of the phase function on the final rendering are partially negligible when operating over thick media dominated by high-order scattering, where after a certain number of bounces, behaves as diffuse (BOUTHORS et al., 2008). However, the effects are perceptually impactful over thin media, where light scatters only a few times before leaving the medium (GKIOULEKAS et al., 2013).

2.6.1 Isotropic

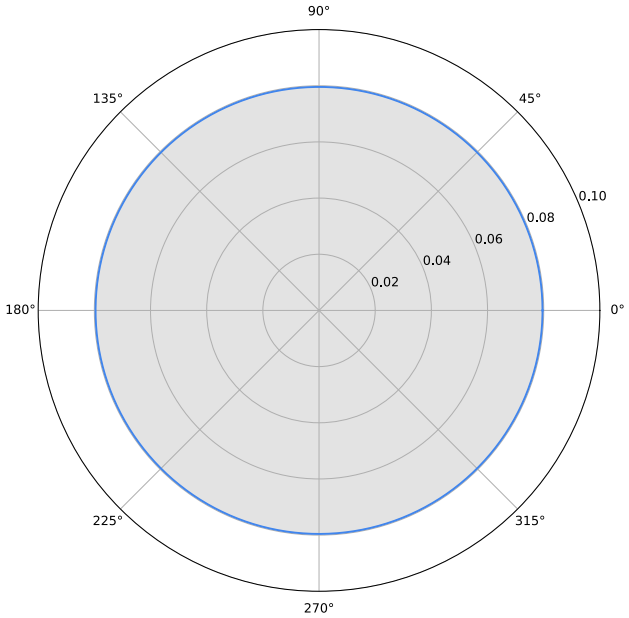
The isotropic phase function follows a uniform sampling of the unit sphere; it can be simply expressed as:

$$f_p(x, \omega, \omega') = \frac{1}{4\pi}$$

where for each given incoming ray the resulting scattering ray will have a random direction with equal probability over the sphere. A polar plot of the 2D isotropic behavior is

shown in Figure 2.8.

Figure 2.8: Isotropic phase function polar plot. It represents a cross section of the spherical distribution of scattered rays.



Source: The Author.

2.6.2 Henyey-Greenstein

The most common phase function used due to its simplicity and flexibility is the Henyey-Greenstein (HG) function. It is able to convey phenomena of backward-scattering, isotropic-scattering and forward-scattering through a single parameter g, ranging from -1 to $+1$. The closer g gets to -1, the more backward-scattering occurs; the closer to +1, the more forward-scattering and $g = 0$ means an isotropic behavior. The HG equation is written as:

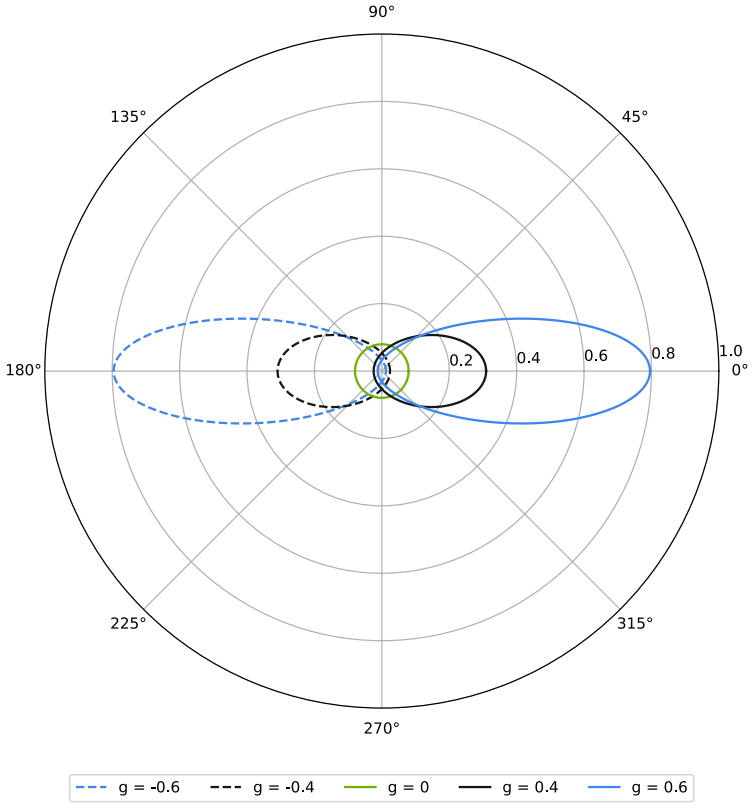
$$f_p(\theta) = \frac{1}{4\pi} \cdot \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)}$$

Alternatively, expressing the phase as a function of ω, we must calculate θ between the scattering ray ω' and the coordinate system frame normal which, assuming we are working with the z-axis as the up vector from the spherical coordinate system, we get a normal axis of $(0, 0, 1)$. Hence, following our notation the formula can be written as:

$$f_p(x, \omega, \omega') = \frac{1}{4\pi} \cdot \frac{1 - g^2}{(1 + g^2 - 2g\cos(\omega' \cdot (0, 0, 1)))}.$$

In Figure 2.9 we plot a polar graph with the effects for different values of g. In Figure 2.10 we showcase the visual impacts of forward- and backward-scattering.

Figure 2.9: The Henyey-Greenstein phase function with backward-scattering, when parameter g is negative, depicted by the dashed line lobes, isotropic when $g = 0$, depicted by the green circle at the center, and forward-scattering, when parameter g is positive and depicted by the solid line lobes on the right.



Source: The Author.

Figure 2.10: The Henyey-Greenstein phase function anisotropy effect. The cloud is lit by an area light source behind it. The cloud on the left shows an effect of high forward-scattering with parameter $g = 0.9$; on the right the cloud shows an effect of high backward-scattering with parameter $g = -0.9$. Note that on the right, due to backward-scattering, most light rays fail to reach the viewer, reflecting backward towards the light source instead, and resulting in a darker cloud.

Source: The Author.

3 RELATED WORK

Volume rendering is a field of computer graphics. One of its lines of research focuses on evolving models for light transport within participating media, i.e. volumes of tiny sparse particles. The different models used to calculate light transport take into account, in order of increasing realism, absorption only, emission only, emission and absorption combined, single scattering of external illumination without shadows, single scattering with shadows, and multiple scattering (Max, 1995).

Throughout the years, several techniques of volumetric rendering have been developed. They can be categorized into six groups, though not entirely since a few of them could fit in more than one group (JÖNSSON et al., 2014). The six groups are based on: local region, slice, light space, lattice, basis function, and ray tracing. These groups are organized taking into account the concepts used to calculate illumination in each model. Moreover, even though all of them are based on Max's foundations (Max, 1995), they can differ significantly. A major difference between them is shadow intensity and quality. Table 3.1 shows all models in each of these groups, which are briefly explained ahead.

Local region based techniques are comparable to the conventional Blinn-Phong illumination model (BLINN, 1977) used for rendering surfaces due to their local shading nature. Models in this group either consider only the current voxel, adjacent voxels, or larger regions, but they are still considered localized regions. Gradient-based volumetric rendering (LEVOY, 1988) explores gradients between voxels as surface normals, similarly calculating light as with the model of Blinn-Phong (BLINN, 1977). Local ambient occlusion (HERNELL; LJUNG; YNNERMAN, 2008) is locally approximated from adjacent voxels. Dynamic ambient occlusion (ROPINSKI et al., 2008) is similar but has a pre-computation stage to generate histograms. This method supports the integration of ambient occlusion, color bleeding, and basic scattering effects.

Slice based techniques use a series of rectangular slices as proxy geometry to portray the volumetric data. Nowadays, with increased memory and GPU support, it is more common to simply map the volumetric data to a 3D texture. Half-angle slicing (KNISS et al., 2003) supports phase functions, spatially varying indirect extinction, multiple scattering, and only one light source. Directional occlusion shading (SCHOTT et al., 2009) does not take into account the emission term and supports only one light source fixed at the camera position. Multidirectional occlusion shading (SOLTÉSZOVÁ et al., 2010) extends the directional occlusion shading by allowing the light source to be positioned

Table 3.1: Volumetric illumination techniques sorted into six groups based on conceptual similarities. Representative papers are listed in parenthesis.

Group	Algorithms
Local region based	• Gradient based (LEVOY, 1988) • Local ambient occlusion (HERNELL; LJUNG; YNNERMAN, 2010) • Dynamic ambient occlusion (ROPINSKI et al., 2008)
Slice based	• Half angle slicing (KNISS et al., 2003) • Directional occlusion (SCHOTT et al., 2009) • Multidirectional occlusion (SOLTÉSZOVÁ et al., 2010)
Light space based	• Deep shadow mapping (HADWIGER et al., 2006) • Image plane sweep (SUNDEN; YNNERMAN; ROPINSKI, 2011) • Shadow splatting (ZHANG; XUE; CRAWFIS, 2005) • Historygram Photon Mapping (JÖNSSON et al., 2012)
Lattice based	• Piecewise integration (HERNELL; LJUNG; YNNERMAN, 2010) • Shadow volume propagation (ROPINSKI; DORING; REZK-SALAMA, 2010) • Summed area table 3D (DíAZ et al., 2010) • Extinction-based shading (SCHLEGEL; MAKHINYA; PAJAROLA, 2011) • Light propagation volumes (ZHANG; MA, 2013) • Discrete ordinates method (DOM) (ELEK et al., 2014)
Basis function based	• Spherical harmonics (KRONANDER et al., 2012)
Ray tracing based	• Monte Carlo ray tracing (KROES; POST; BOTHA, 2012)

Source: Compilated by (JÖNSSON et al., 2014).

arbitrarily in the viewer's hemisphere.

Light space based techniques consider that illumination is directionally propagated concerning the current light position, but because of that, these techniques usually support only one light source. Deep shadow mapping (HADWIGER et al., 2006) stores shadow information as a data structure of transmittance but supports only one light source. Image plane sweep volume illumination (SUNDEN; YNNERMAN; ROPINSKI, 2011) supports one point or directional light source and single and multiple forward-scattering. Shadow splatting (ZHANG; XUE; CRAWFIS, 2005) supports multiple point light sources, single scattering, and uses Phong shading. Histogram photon mapping (JÖNSSON et al., 2012) supports multiple scattering and multiple light sources arbitrarily positioned in the environment.

Lattice based techniques directly use the grid to calculate illumination per voxel. Shadow volume propagation (ROPINSKI; DORING; REZK-SALAMA, 2010) distributes

illumination in a pre-processing phase, storing its calculation in a volumetric data grid. It supports multiple scattering and only one point or directional light source in any arbitrary positions, with low frame rates because of its additional lookup in the extra 3D texture. Piecewise integration (HERNELL; LJUNG; YNNERMAN, 2008) supports only one directional light source, approximates in-scattering as emission, and usually uses ray casting. Extinction-based shading and illumination (SCHLEGEL; MAKHINYA; PAJAROLA, 2011) uses summed area tables (SAT) to compute extinction and supports point and directional light sources. Light propagation volumes (ZHANG; MA, 2013) support multiple point and directional light sources and both single and multiple isotropic scattering at interactive frame rates. The discrete ordinates method (ELEK et al., 2014) supports multiple anisotropic scattering in heterogeneous participating media, but it is only physically plausible, and is limited to low grid resolutions.

Basis function based techniques use basis functions to represent light source radiance and transparency. Spherical harmonics (KRONANDER et al., 2012) uses a data structure similar to a mipmap to store multiple resolutions of the volumetric dataset, thus requiring a lot of storage. Both area and point light sources are supported.

Figure 3.1: Results obtained with one representative method from each category. From left to right: gradient-based shading (LEVOY, 1988), directional occlusion shading (SCHOTT et al., 2009), image plane sweep volume illumination (SUNDEN; YNNERMAN; ROPINSKI, 2011), shadow volume propagation (ROPINSKI; DORING; REZK-SALAMA, 2010), and spherical harmonics lighting (KRONANDER et al., 2012).

Source: A survey of volumetric illumination techniques for interactive volume rendering (JÖNSSON et al., 2014).

More recent works in the industry of video games (HILLAIRE, 2015b; SHNEIDER, 2015; SHNEIDER, 2017; ENGEL, 2016; BAUER, 2019), which are directly focused on real-time rendering, have achieved better results regarding real-time performance. Clouds in the "Horizon Zero Dawn" game use ray marching (SHNEIDER, 2017)

with single scattering only, and it reaches a performance of 2ms per frame (ENGEL, 2016). In the same year, Hillaire also proposed a real-time approach for single scattering with temporal volumetric integration, reaching performance of 2.95ms per frame (HILLAIRE, 2015b). Even today, we are still limited to single scattering in high-performance real-time volumetric rendering, as evidenced in games and academic works. Two such examples are the results in "Red Dead Redemption 2", presented in Figure 3.2, and the Siggraph course (BAUER, 2019) "Advances in Real-Time Rendering in Games."

Figure 3.2: Volumetric clouds in "Red Dead Redemption 2", which is built upon the base created by Andrew Schneider and other technologies developed throughout the industry over the years.

Source: (BAUER, 2019).

Multiple scattering is a factor that plays a huge role into physically rendering participating media with high albedo. A good quality image synthesizer should also take into account anisotropic scattering effects for media with high albedo, such as clouds. Older methods account only for single scattering (LEVOY, 1988; HERNELL; LJUNG; YNNERMAN, 2010; ROPINSKI et al., 2008) and some with limitations to isotropic scattering behavior. Recent work started taking into account anisotropic multiple scattering for interactive rendering, but with limitations on grid size (ZHANG; MA, 2013; KRONANDER et al., 2012). Some work do scale well with grid size and take into account multiple scattering, but do not scale well with image size (ZHANG; XUE; CRAWFIS, 2005; HADWIGER et al., 2006).

A few other recent works, that were not available at the time of Jönsson et al. categorization, and published concomitantly to the development of our work, are worth mentioning. Hofmann et al. propose a temporally stable neural denoiser (HOFMANN et al., 2021), that produces results in real time. However, at a resolution of 1920×1080

they were still limited to 16 frames per second using an RTX 3090 to render the Disney cloud dataset (DISNEY, 2021). On thin details, the denoiser causes a blur effect that is greatly noticeable when near the cloud volume. Furthermore, Miller et al. (MILLER; GEORGIEV; JAROSZ, 2019) improve upon previous work directed at offline rendering by proposing a null-scattering path integral formulation of light transport (MILLER; GEORGIEV; JAROSZ, 2019) which, although not focused on real-time rendering, lays the foundation for state-of-the art volumetric rendering, thus making possible to generate ground-truth images used for validation.

Although many of these methods have improved volumetric rendering, each one using different types of techniques to achieve its results, nobody has yet used the CIEE metric as a way to speed computation and make visual comparative visual analysis, as we propose in Chapter 5.

4 VOLUMETRIC RENDERING ALGORITHMS

Rendering algorithms for volumes come in many flavors. Each algorithm tries to solve the rendering equation by setting constraints that meet their needs, be it time or quality restrictions. In this chapter, we discuss a few approaches and their use cases, focusing more on null-collision algorithms, which is what we chose and implemented in our own in-house *Narval Engine* for validation and ground truth purposes.

4.1 Distance and Transmittance Sampling

Distance sampling is the process of sampling the flight a photon travels until its next interaction occurs within the participating medium.

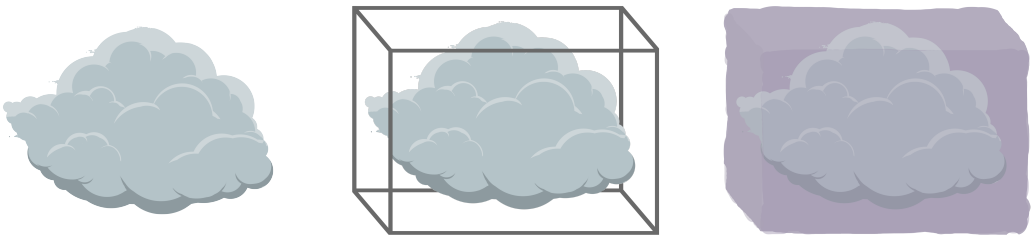
There are three types of approaches to find the distance t: analytic, semi-analytic, and approximate. The analytic closed-form tracking is used in simple homogeneous volumes where the extinction σ_t is constant and an inverted Cumulative Distribution Function CDF^{-1} can be used to sample transmittance along a path. It is commonly applied to render fogs and atmospheres. Semi-analytic approaches are used in piece-wise and heterogeneous volumes. The category of null-collision algorithms, which are semi-analytic, is discussed further in Section 4.2. The most common approach used in real-time applications is ray marching, which is approximate, due to its simplicity and high performance. It uses a fixed step to find the distance collision through the volume. Its major downside is the introduction of bias and tendency to have detail losses more visible in thin media.

4.2 Null-collision Algorithms

This class of algorithms are semi-analytic and unbiased. Originating from neutron transport and plasma physics, it was first applied to rendering by (RAAB; SEIBERT; KELLER, 2008). This is the most common class of algorithms chosen when working with heterogeneous media. For distance sampling, i.e., free path sampling, there are four major algorithms: delta tracking, weighted delta tracking, decomposition tracking and spectral tracking. We chose to implement delta tracking, the same method employed in *pbrt v3* (PHARR; JAKOB; HUMPHREYS, 2016). For transmittance sampling there are four major algorithms: delta tracking, ratio tracking, residual ratio tracking, and next-

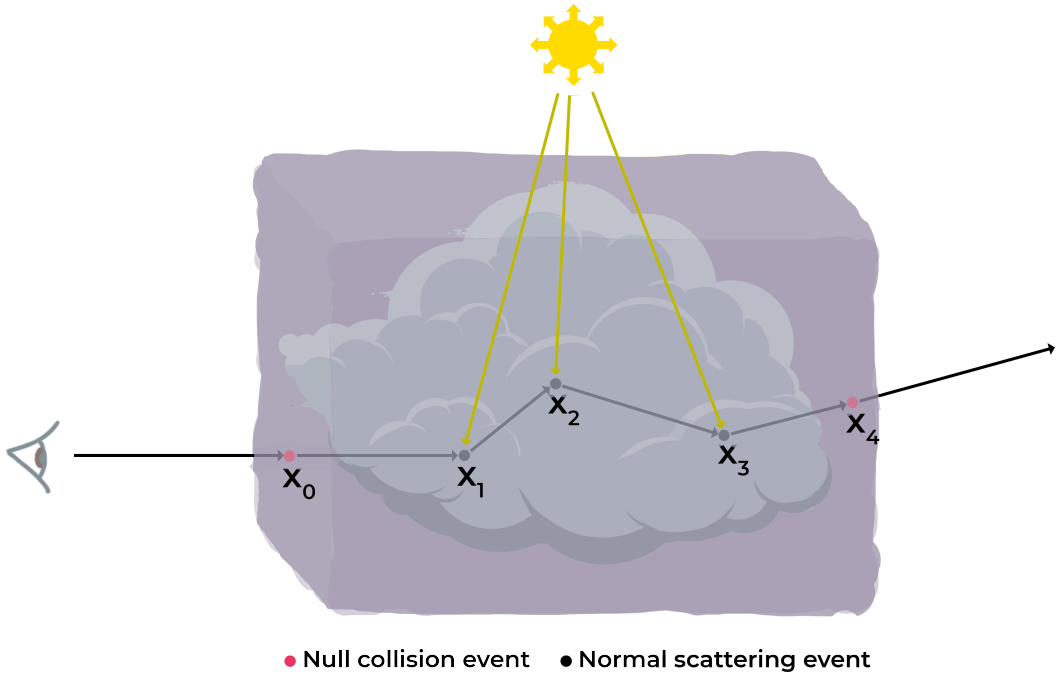
flight delta/ratio tracking. We opted for ratio tracking, which is also the same method employed in *pbrt v3*. The key concept in null-collision algorithms is the introduction of a fictitious medium that fills all empty areas inside the bounding box of a volume. This idea is illustrated in Figure 4.1 by the purple filling around the cloud. While introducing this fictitious medium we also introduce the null-collision event, which is a collision where we do not make any light transport calculation and keep the ray direction unchanged, as illustrated in Figure 4.2 with red points.

Figure 4.1: Visual interpretation of the fictitious matter where null collisions may occur. Note that we treat all empty areas within the volume bounding box as fictitious matter, represented here in light purple.



Source: The author.

Figure 4.2: For each null-collision event that happens, the path direction remains unchanged until a normal scattering event happens inside the proper volume. The sample points x_0 and x_4 represent a null-collision event, and x_1, x_2, and x_3 a normal scattering event.



Source: The author.

4.2.1 Delta Tracking

Delta tracking is an algorithm used to sample medium interactions. The general idea of delta tracking, also known as Woodcock tracking, the null-collision algorithm, or pseudo scattering, is to homogenize the volume with a fictitious density by creating a new collision type. In this new type of collision, called null-collision and represented by the coefficient σ_n, the direction of the ray is preserved, not affecting light transport, as shown in Figure 4.3. In order to have the volume homogenized, σ_n must be chosen respecting the restriction that the free path coefficient $\tilde{\sigma}$ becomes constant.

$$\tilde{\sigma} = \sigma_a + \sigma_s + \sigma_n.$$

In this method we now have three collision types to account for as probabilities:

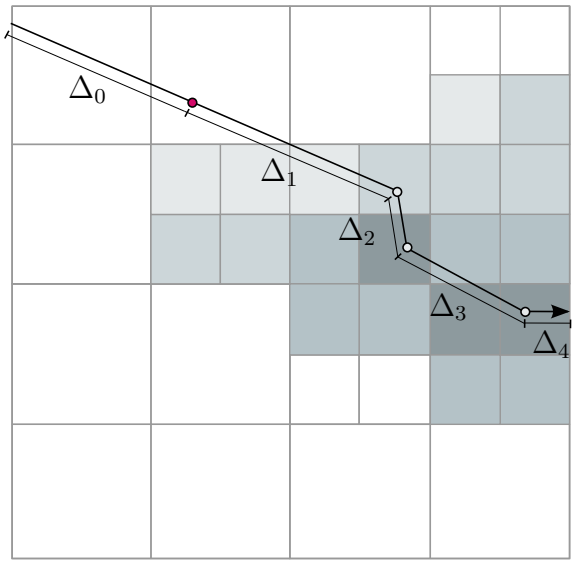
$$P_a(x) = \frac{\sigma_a(x)}{\tilde{\sigma}}, \quad P_s(x) = \frac{\sigma_s(x)}{\tilde{\sigma}}, \quad P_n(x) = \frac{\sigma_n(x)}{\tilde{\sigma}}.$$

Adapting the VRE to support null-collision probability gives the following equation:

$$L(x,\omega) = \int_{t=0}^{\infty} [P_a(x)L_e(x,\omega) + P_s(x)L_s(x,\omega) + P_n(x)L(x,\omega)]dt.$$

Interpreting the mathematical formulation and translating it into pseudo code gives Algorithm 1.

Figure 4.3: Distance sampling at a microscale level. Starting at the volume boundary, each sampled distance gives a point along the current ray direction and, if this point is inside the medium, and thus results in a valid normal scattering event, we move on to sample the phase function for a new ray direction. We repeat this step until we reach the boundary again or a transmittance value of 0, i.e. an obstacle or an absorbed ray.



Source: The author.

4.2.2 Ratio Tracking

The delta tracking based estimator of transmittance can be interpreted as a random walk terminated by russian roulette. An unbiased transmittance estimator that is related to delta tracking was introduced to computer graphics by (NOVáK; SELLE; JAROSZ,

Algorithm 1: Delta tracking. x is the sampled point, ω is the ray direction, and d is the sampled distance.

procedure deltaTracking(x, ω, d)**:**
- **while** *true* **do**
  - $\xi \in [0, 1]$
  - $t = -\frac{ln(1-\xi)}{\tilde{\sigma}}$
  - **if** *t* > *d* **then**
    - **return** $L_d(x, \omega)$
  - **end**
  - $x = x - t \times \omega$
  - **if** $\xi < \frac{\sigma_a(x)}{\tilde{\sigma}}$ **then**
    - **return** $L_e(x, \omega)$
  - **else if** $\xi < 1 - \frac{\sigma_n(x)}{\tilde{\sigma}}$ **then**
    - $\omega = sample \propto phase(x, \omega)$
    - $d = new\_Ray\_End\_In\_\omega$
  - **else**
    - $d = d - t$
  - **end**
- **end**

2014). The russian roulette of delta tracking is replaced by a Monte Carlo weight that is equal to the probability of a null collision in relation to the extinction coefficient at a given sampled point. The ray is navigated backwards while updating the weight until the boundary is hit. This method is called ratio tracking and a pseudo code is given in Algorithm 2.

Algorithm 2: Ratio tracking. x is the sampled point, ω is the ray direction, and d is the sampled distance.

procedure ratioTracking(x, ω, d)**:**
- $Tr = 1$
- **while** *true* **do**
  - $\xi \in [0, 1]$
  - $t = -\frac{ln(1-\xi)}{\tilde{\sigma}}$
  - **if** *t* > *d* **then**
    - **return** Tr
  - **end**
  - $x = x_0 - t \times \omega_0$
  - $Tr = Tr \times (1 - \frac{\sigma_t(x)}{\tilde{\sigma}})$
- **end**

4.3 Volume as Material

Once the methods used to sample distance and transmittance of ray paths inside a volume are chosen, the next step is to integrate these algorithms into the path tracing equation. The *Narval Engine* works with two file formats when importing volumes: vol and vdb. The format .vol is a simple text file containing its resolution in the first line, followed by the grid density as an array of floating-point numbers written as plain text, where each row represents a row into the matrix. The second format .vdb, from the OpenVDB project (FOUNDATION,), which is an open source initiative held and maintained by the Academy Software Foundation (ASWF).

The volume material properties are parameterized into: absorption, scattering, density multiplier, and phase function. Absorption corresponds to a vector of 3 floating-point numbers that represents the volume absorption in m^{-1} for each RGB color channel. Scattering corresponds to a vector of 3 floats that represents the volume probability of a scattering event per m^{-1}. The density multiplier is a constant factor that represents how thin or thick the volume is. As this value gets larger the volume starts to appear like a solid, with dense matter. The last parameter is the phase function, either isotropic or Henyey-Greenstein. An example scene with a cloud centered at origin and an area light emitter behind it is given in Listing 4.3. Every scene descriptor is a JSON file and mainly composed of four areas of configuration: materials, primitives, camera, and renderer. For this scene we first define both a light emitter and cloud materials. The emitter material is set by attributing a name to the material, its type ("emitter"), and its albedo (radiance). The cloud must also have a name and type, which is "volume" followed by scattering, absorption, phase function, path for its file relative to the executable and density multiplier. Next we create two primitives, one rectangle for the light emitter and a volume for the cloud, which is internally treated as an Axis Aligned Bounding Box (AABB). For each primitive one must define its transformations with corresponding position, scale, and rotation (in degrees). It is also necessary to specify the material name to which we will attribute the primitive to. The camera is defined by its position, point to which look at, up vector, movement speed in meters per second, vertical field of view (FOV), aperture (functional only when in offline rendering mode), and focal distance. The renderer is defined by its resolution, number of samples per pixel (SPP), number of bounces per camera ray, and whether HDR and tone mapping is active or not.

```
{
	"materials": [
		{
			"name": "emitter",
			"type": "emitter",
			"albedo": [20, 20, 20]
		},
		{
			"name": "cloudVolMat",
			"type": "volume",
			"scattering": [1.1, 1.1, 1.1],
			"absorption": [0.01, 0.01, 0.01],
			"phaseFunction": "hg",
			"g": 0.9,
			"path" : "vdb/wdas_cloud_quarter.vdb",
			"density": 10
		}
	],

	"primitives": [
		{
			"name": "rectLight",
			"type": "rectangle",
			"transform": {
				"position": [0, 0, 80],
				"scale": [150, 100, 1],
				"rotation": [0, 0, 0]
			},
			"materialName": "emitter"
		},
		{
			"name": "cloud",
			"type": "volume",
			"transform": {
				"position": [0.0, 0, 0],
				"scale": [15.9, 9.51, 13.5],
				"rotation": [0, 0, 0]
			},
			"materialName": "cloudVolMat"
		}
	],

	"camera": {
		"position": [0, 0, -17],
		"lookAt": [0, 0, 0],
		"up": [0, 1, 0],
		"speed": 5,
		"vfov": 45,
		"aperture": 0.0001,
		"focus": 1
	},
```

```
        "renderer": {
                "resolution": [1200, 600],
                "spp": 1000,
                "bounces": 3,
                "HDR": true,
                "toneMapping": true
        }
}
```

Listing 4.1: Example of a scene file description.

5 IMAGE ANALYSIS AND VALIDATION

We investigate our novel approach to volumetric illumination that allows approximation of multiple scattering in real time. We first lay a foundation analysis to see how the Samples Per Pixel (SPP) and Bounces parameters affect our renderer's speed and quality. Next, with the ground truth computed and perceptual analysis done, we proceed to discuss the novel algorithm in Chapter 6.

Our approach to achieve better performance is inspired by the technique presented by (KRAVCHENKO et al., 2017), applying the CIELAB metric (MAHY; EYCKEN; OOSTERLINCK, 1994) to quantitatively compare results to a ground truth, and to determine which parameters improve performance without losing perceived quality.

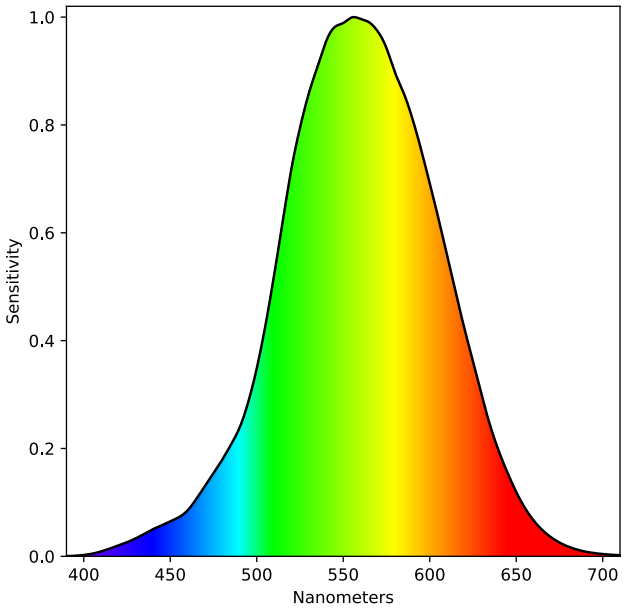
The human visual system is more sensible to certain wavelengths, more precisely the ones near 555nm, making yellow-greenish hues more noticeable (PEGORARO, 2016). In Figure 5.1 one can visualize this increased sensitivity near wavelengths at 555nm. Since certain color changes are less noticed, we can reduce processing demand by scaling down parameters in our algorithms, while monitoring the significant changes for the human eye.

This concept is conveniently encapsulated when working in the CIELAB color space, also known as L*a*b*. It represents colors as a triplet (L*,a*,b*) where L* stands for perceptual lightness, and a* and b* the variation of four unique colors: red, green, blue and yellow. A visual representation of this color space is given in Figure 5.2.

The CIELAB similar images metric, states that two images will look similar to a standard observer if they have a just noticeable difference (JND) (SHARMA, 2003) of $\Delta E^*_{ab} \approx 2.3$. The variable ΔE^*_{ab} is defined as the Euclidean distance between two images in the CIELAB color space (MAHY; EYCKEN; OOSTERLINCK, 1994). This distance is computed by Equation 6, where L^*, a^*, and b^* are the color axes in the CIELAB color space illustrated in Figure 5.2.

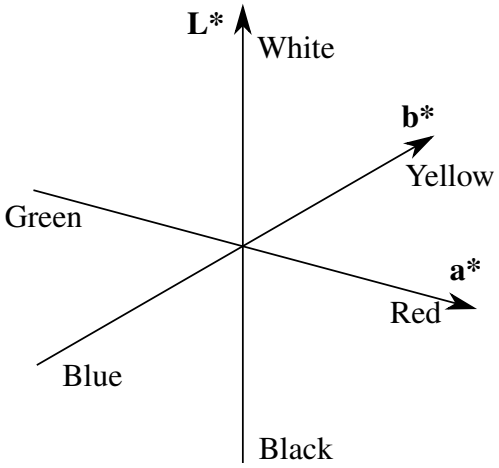
$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \tag{6}$$

Figure 5.1: Photopic brightness sensitivity of the human eye as a function of wavelength.



Source: The author.

Figure 5.2: CIELAB Color Space Axes. L^* varies from 0 to 100, b^* and a^* varies from -128 to 128.



Source: The author.

In their work, Kravchenko et al., propose a technique to bring the iridal light transport (ILIT) model, previously presented in (LAM; BARANOSKI, 2006), up to interactive frame rates. To achieve this goal, they use CUDA to adapt the model for GPU processing and adapt their framework parameters, such as rays and spectral resolution, to achieve better performance while respecting the CIELAB metric. Their ground truth, used as reference, was generated with a high spectral resolution and thousands of rays up to a point of high fidelity biophysically-based iris. In a similar manner, we propose to generate our

ground truth reference with as high as possible quality, with thousands of rays and ray bounces. Our first approach is to reduce the quantity of rays and bounces maintaining interactive frame rates and meeting the CIELAB threshold.

We use *pbrt v4* renderer from Matt Pharr et al. (PHARR; JAKOB; HUMPHREYS, 2016) as a reference starting point to validate our path tracer. The choice of version 4, instead of version 3, was made based on the fact that the more recent version 4 of *pbrt* encompasses newer and more accurate algorithms for volumetric rendering. The *Narval Engine* uses delta tracking to sample medium interactions and ratio tracking to estimate transmittance, whereas *pbrt v4* is based on the null-scattering path integral formulation (MILLER; GEORGIEV; JAROSZ, 2019). The resulting comparison is shown in Figure 5.3, where differences between images mainly appears in shadow areas, still keeping general shape and detail. Since we do not use the same volumetric rendering algorithm as *pbrt v4*, the results are not identical. Because of limitations and other implementation nuances, we are not able to perfectly align the images and thus cannot use RMSE for a statistical comparison. Nevertheless, it reassures us that we do not have any significant deviations or artifacts in our path tracer.

Figure 5.3: Disney cloud dataset one quarter size, rendered with our *Narval Engine*, as shown on the left side, and with *pbrt v4*, as shown on the right side, respectively. Both images were generated with 10^4 SPP.

Source: The author.

All tests were computed on a laptop equipped with an RTX 3060 GPU with 6 GB of VRAM, an Intel i7-10870H and 16 GB of RAM. The GPU renderer is coded using OpenGL shaders and the CPU renderer coded using multi-thread tiled rendering. For all tests in this chapter the GPU version was used in order to have the best performance possible.

Next, we make a psychophysical analysis using the CIE JND metric upon some parameters of our volumetric rendering, mainly the number of ray samples and bounces per pixel. Our goal is to relate visual response to stimulus intensities, making subjective

perception measurable and relating physical and psychological scales through the Just Noticeable Difference (JND) from CIELAB. To improve results and retain as much illumination detail as possible, all images were generated and treated in EXR format, which has 32 bits per channel in linear float space, before the final conversion to PNG, which are then used as figures in this dissertation.

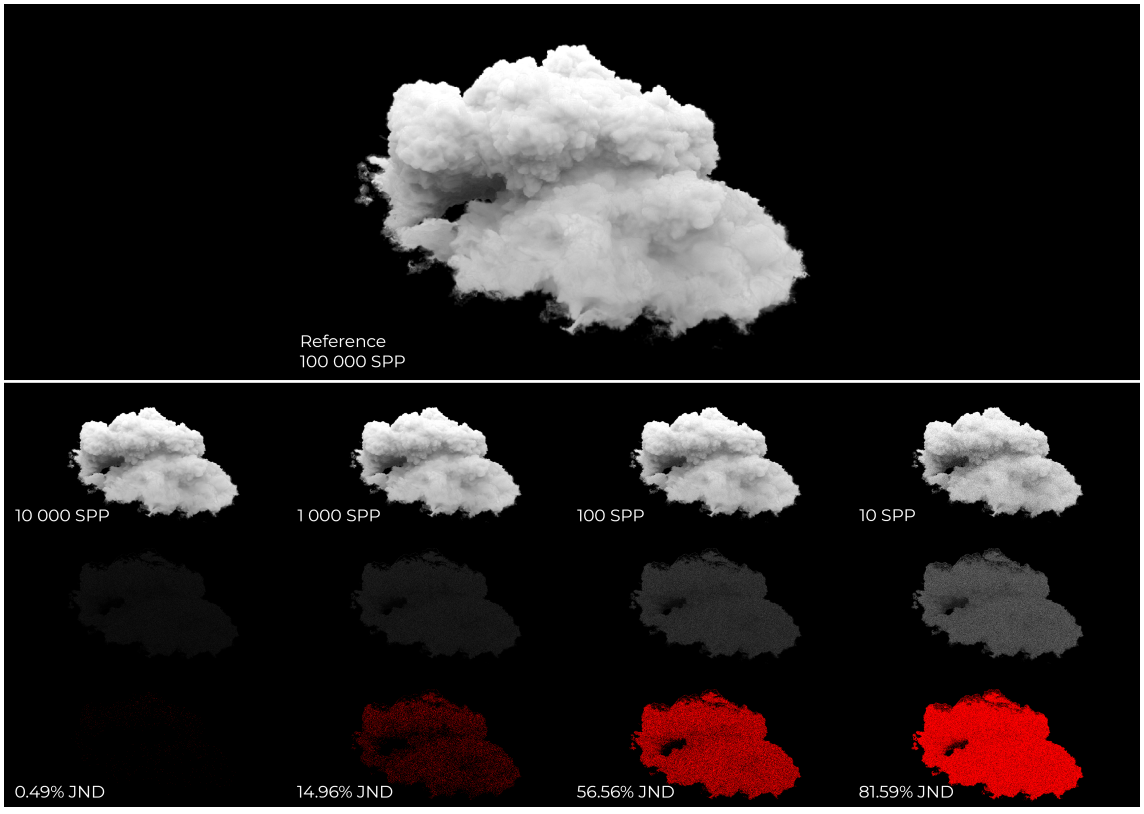
We first analyze how the number of samples per pixel (SPP) affects the JND metric. For each test, five clouds were rendered with decreasing numbers of SPP, starting with 10^5 SPP, denoted as reference, followed by 10^4 SPP, 10^3 SPP, 100 SPP, and 10 SPP. These results are shown in Figure 5.4 and Figure 5.5. The first test shown in Figure 5.4 was conducted with the number of bounces limited to a maximum of 3 and the second test shown in Figure 5.5 limited to a maximum of 50 bounces. For each rendered image we compare it against the reference. In Figure 5.4 and Figure 5.5 the first row shows the reference, the second row shows each subsequent image to be compared against the reference, the third row is the CIELAB ΔE difference between the column rendered image and the reference in its column, and the fourth row shows in red the pixels that exceed the CIELAB Just Noticeable Difference (JND) metric of $\Delta E \approx 2.3$.

For the first test, shown in Figure 5.4, the differences between 10^5 SPP and 10^4 SPP are negligible, where only 0.49% pixels exceed the JND, thus the overwhelmingly majority of pixels being visually identical to a human observer. Although the visually perceptual difference is almost null, the reference took approximately 137 minutes, whereas the one with 10^4 SPP took only 13 minutes, a difference of $10.5\times$ longer. Similarly, when comparing the reference against 10^3 SPP, the JND was reasonably low, at 14.96%. Timewise, the difference, following a linear pattern, between the reference and 10^3 SPP was $100.9\times$ longer, from 137 minutes to 1 minute. The next two JND trials of 100 SPP and 10 SPP against the reference were too high, at 56.5% and 81.5%, respectively. All rendering times are presented in Table 5.1. In Figure 5.6, we zoom on the tail of the cloud, where the thin media display a more pronounced divergence against the reference, thus highlighting areas where the difference is more acute.

For the second test, shown in Figure 5.5, the differences between 10^5 SPP and 10^4 SPP are still negligible, where only 2.24% pixels exceed the JND. Although the visual perceptual difference is not significant, when operating with 50 bounces the reference took much longer, approximately 354 minutes, against 137 minutes when operating with only 3 bounces. The same linear time pattern applies when operating with 50 bounces, for each time we divide the number of SPP by 10, the time it takes to render also drops by a

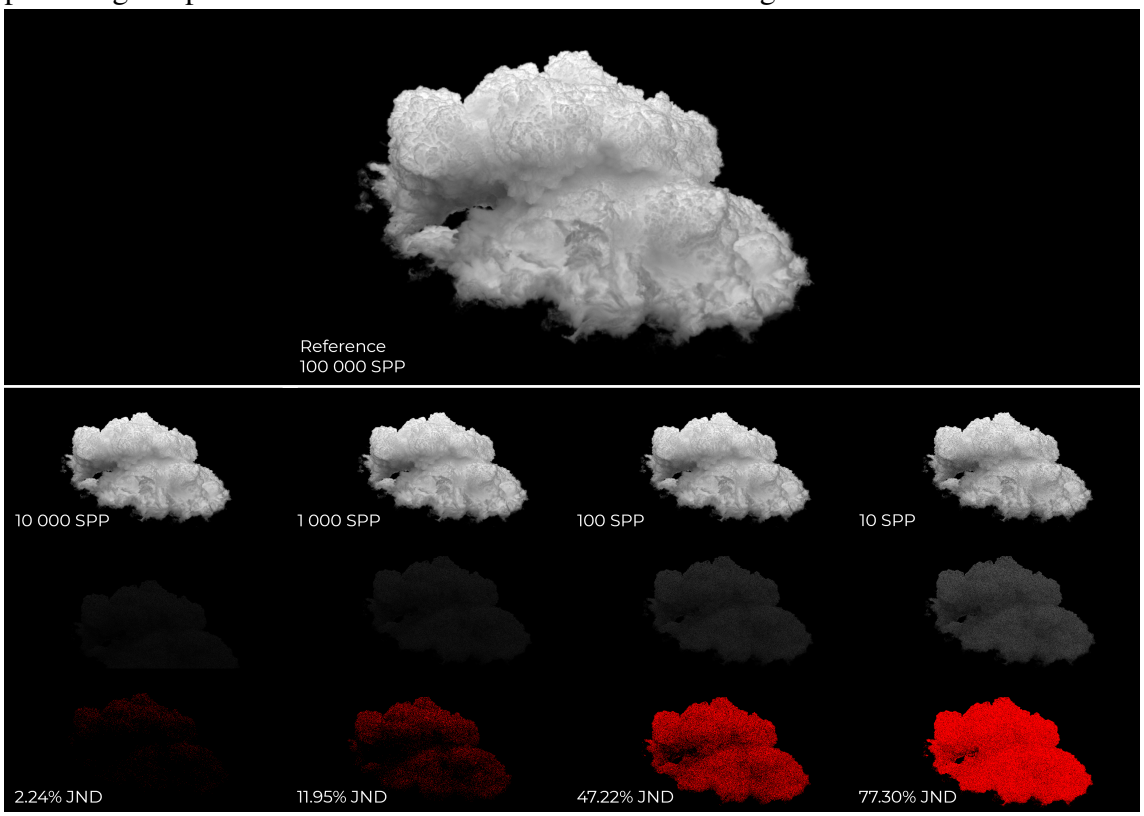
factor of 10. When comparing the reference against 10^3 SPP, the JND was reasonably low, at 11.95%. Time-wise, the difference, following a linear pattern, between the reference and 10^3 SPP was $106.5\times$ longer, from 354 minutes to 3 minutes. The next two JND trials of 100 SPP and 10 SPP against the reference were too high, at 47.22% and 77.3%, respectively. All rendering times are presented in Table 5.1.

Figure 5.4: Disney Cloud dataset at one quarter its original size, rendered with 10^5, 10^4, 10^3, 10^2, and 10 Samples Per Pixel (SPP), respectively, and limited to a maximum of 3 bounces. The first row shows the reference with 100 000 SPP, the third row shows the ΔE difference between the upper image in the same column and our reference, the first cloud rendered with 10^5 SPP. The fourth row shows pixels exceeding the CIELAB Just Noticeable Difference (JND) metric of $\Delta E \approx 2.3$, shown as red pixels alongside the total percentage of pixels that exceed the JND for the whole image.



Source: The author.

Figure 5.5: Disney Cloud dataset at one quarter its original size, rendered with 10^5, 10^4, 10^3, 10^2, and 10 Samples Per Pixel (SPP), respectively, and limited to a maximum of 50 bounces. The first row shows the reference with 100 000 SPP, the third row shows the ΔE difference between the upper image in the same column and our reference, the first cloud rendered with 10^5 SPP. The fourth row shows pixels exceeding the CIELAB Just Noticeable Difference (JND) metric of $\Delta E \approx 2.3$, shown as red pixels alongside the total percentage of pixels that exceed the JND for the whole image.

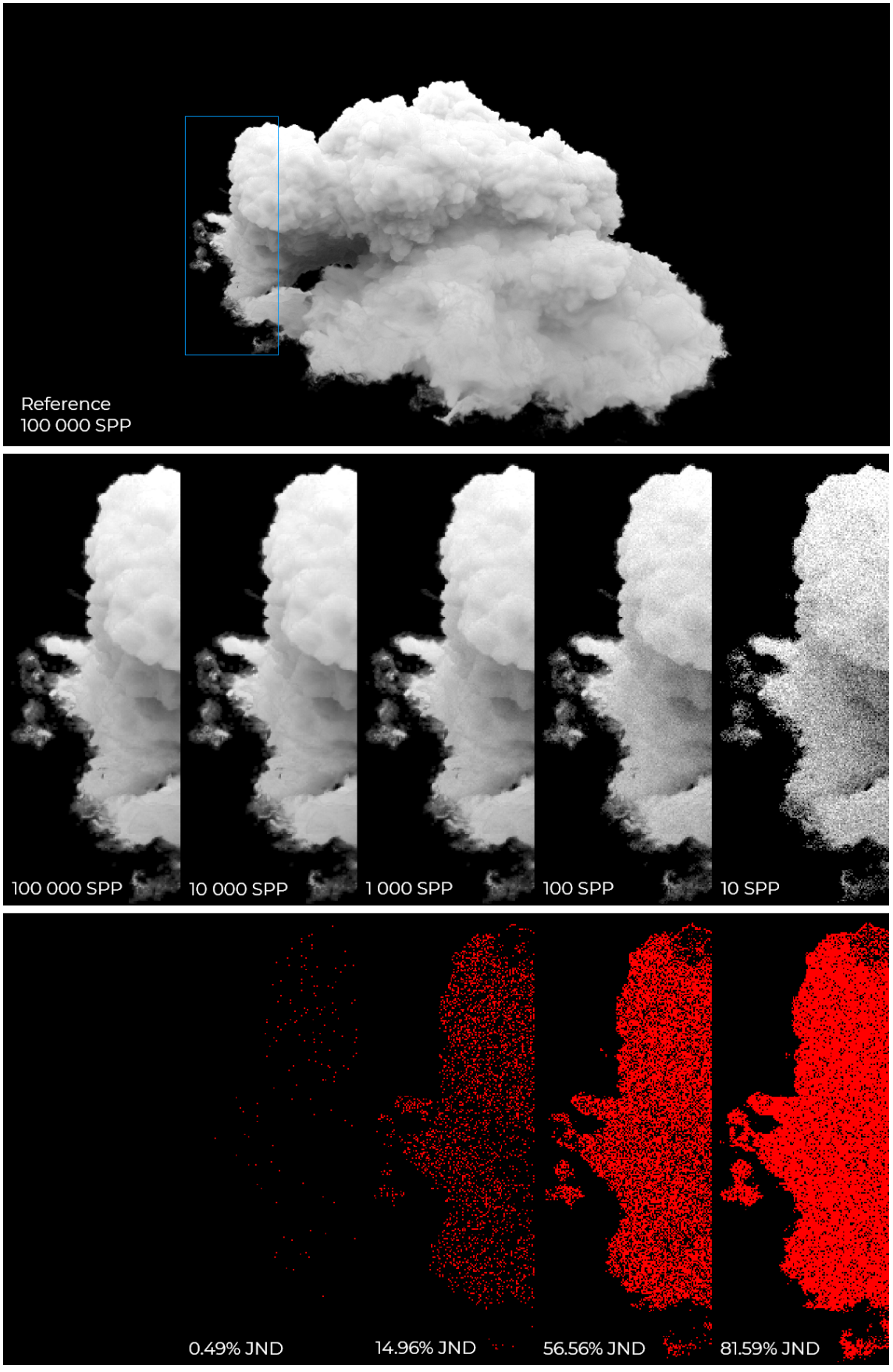


Source: The author.

Table 5.1: Rendering time in seconds for each SPP parameter value used in Figure 5.4 and Figure 5.5.

	3 bounces		**50 bounces**	
SPP	**Time (seconds)**	**Speed up**	**Time (seconds)**	**Speed up**
100 000	8235.6	—	21279.6	—
10 000	819.3	10.05	2026.2	10.50
1 000	81.6	10.04	199.7	10.14
100	8	10.20	19.5	10.24
10	0.8	10.00	1.7	11.47

Figure 5.6: Samples Per Pixel experiment from Figure 5.4 zoomed on the tail of the cloud where we have thin details that show less convergence, thus more prone to noise. The second row shows the final rendered image and the third row shows the JND between the cloud above in the same column and the reference cloud with 100 000 SPP.



Source: The author.

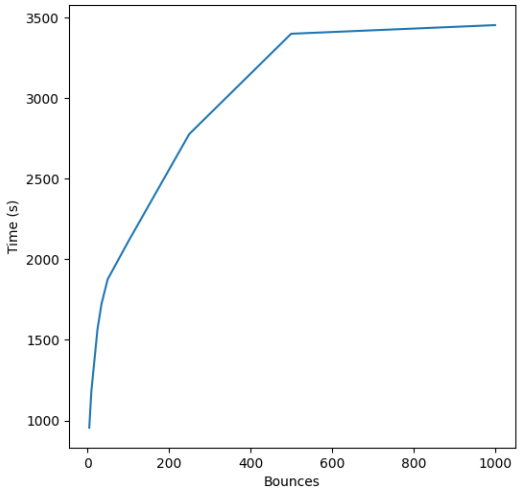
The next step in our analysis is to evaluate how the number of bounces affects image quality and how the human eye perceives it. All images were fixed at 10^4 SPP. We start with 1000 bounces, denoted as reference, and decrease them to 100, 50, 35, 25, 10, and 5 bounces, respectively. These results are shown in Figure 5.8. As with our first experiment the first row shows the reference, the third row is the CIELAB ΔE difference between the rendered image in the same column and the reference, and the fourth row shows in red the pixels that exceed the CIELAB Just Noticeable Difference (JND) metric of $\Delta E \approx 2.3$. The increase in bounces allows the participating medium to accumulate all the energy that enters the medium and disperses through it, resulting in less attenuated shadows than when using a lower number of bounces. This is emphasized in Figure 5.9 when zooming in near the cloud tail. However, increasing the number of bounces has one consequence that an artist has to take into account: the number of bounces increases very rapidly the time taken to render until it reaches an equilibrium, as shown in Table 5.2 and in Figure 5.7. This equilibrium happens as a consequence of either the transmittance reaching 0, when all light gets absorbed during this light path, or when the ray escapes the volume boundaries before reaching the parameter value set as maximum number of bounces.

The number of bounces affects the JND mostly in areas of stronger shadowing, as expected, and is shown in the last row of Figure 5.9. The difference between the reference with 1000 bounces and 100 bounces is only 0.82% JND, which is imperceptible. On the other hand as we decrease from this value, changes escalate quickly. Decreasing to 50 bounces the difference becomes 6.46% JND and decreasing them by half, to 25 bounces, the JND more than triples, reaching 23.05% JND.

Table 5.2: Rendering time in seconds for each number of Bounces used in Figure 5.8.

Bounces	**Time (seconds)**
1000	3452.9
100	2107.8
50	1875.6
35	1724.1
25	1564.4
10	1178.7
5	955.7

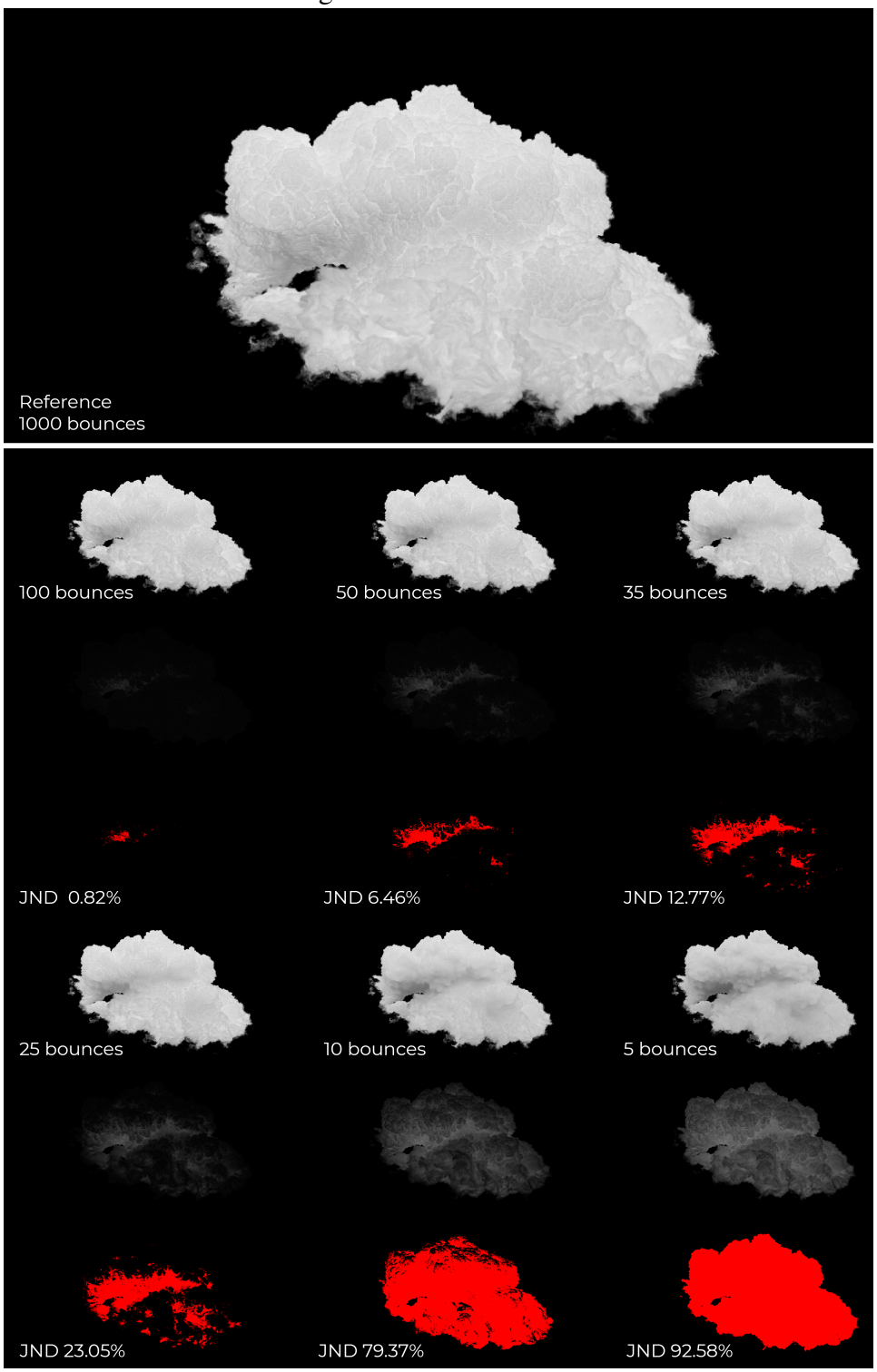
Figure 5.7: Time taken to render a participating medium with increased numbers of bounces grows very rapidly.



Source: The author.

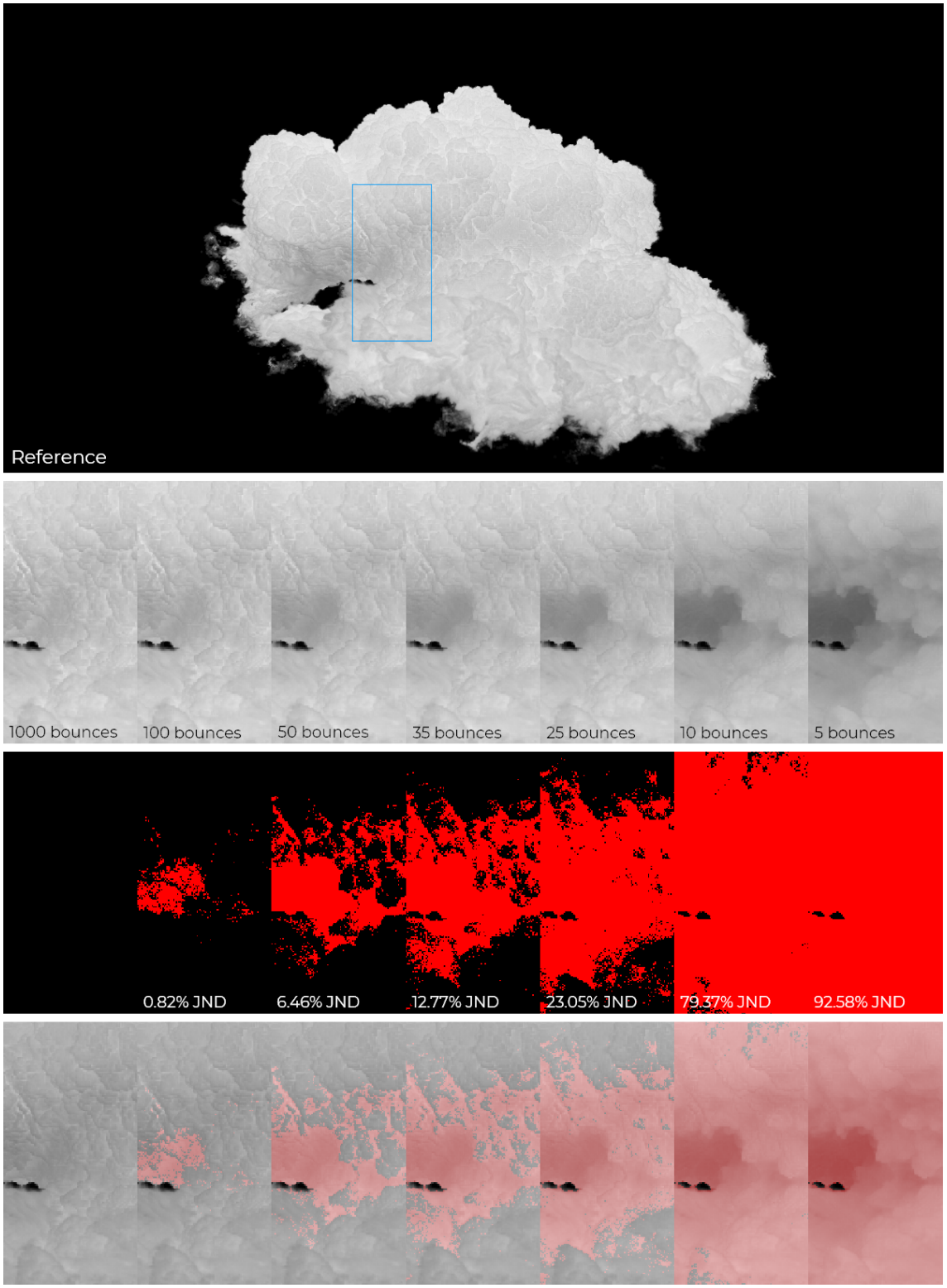
In conclusion, we recommend a minimum of 10^4 SPP and 50 bounces if the goal is to have a physically plausible result. For a more fluffy cloud an artistic choice of less than 10 bounces is also feasible, at the cost of neglecting image fidelity, but gaining in performance.

Figure 5.8: Disney Cloud dataset at one quarter its original size, rendered with fixed 10^4 Samples Per Pixel (SPP) and 1000, 100, 50, 35, 25, 10, and 5 bounces, respectively. The second row under each cloud shows the ΔE difference between the rendered cloud and our cloud with largest number of bounces, the first cloud is rendered with 1000 bounces. The third row shows pixels exceeding the CIELAB Just Noticeable Difference (JND) metric of $\Delta E \approx 2.3$ shown as red pixels alongside the total percentage of pixels that exceed the JND for the whole image.



Source: The author.

Figure 5.9: Bounces experiment zoomed in an area where shadows are more pronounced. The first row shows the final rendered image, the second row shows the JND between the cloud in its column and the reference, and the last row shows an overlay between the first row and second row, highlighting areas where the JND exceeds 2.3.



Source: The author.

6 LOBE ESTIMATOR

In the previous chapter we discussed how to increase performance by reducing SPP and bounces while maintaining perceptual visual quality. However, it is still costly to render using path tracing, even when using lower numbers of SPP and bounces, thus requiring a new method, capable of real-time rendering.

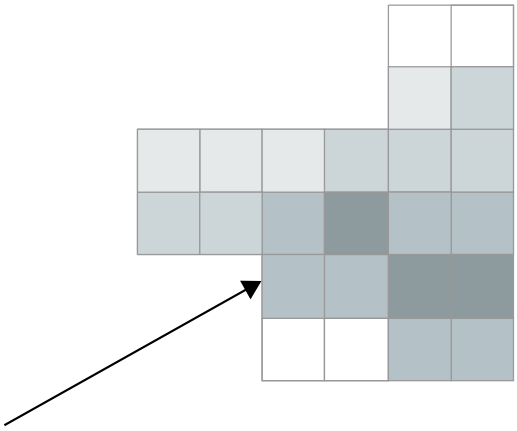
To achieve real-time rendering of participating media and compete with the most common technique used nowadays in real-time for this purpose, ray marching, we propose to investigate a new method, Lobe Estimator. Given the importance of the phase function (GKIOULEKAS et al., 2013) and how multiple-scattering contributes and behaves diffusely after a certain number of bounces (BOUTHORS et al., 2008), we empirically started by testing sampling points along the directions of the phase function instead of along the ray. Therefore, we propose a new method called Lobe Estimator.

The Lobe Estimator algorithm starts by casting a ray into the volume until it reaches a non-empty voxel. The whole process is illustrated step by step in Figure 6.1 and formalized in Equation 7. Once we have the collision point in this voxel we cast another ray, in the same direction as the previous one, at a depth of t, defined by the mean path of the volume, $\frac{1}{\sigma_t}$, multiplied by a depth factor. With this new point inside the volume we sample the phase function randomly for directions and shape them along the ray direction axis, and use this point as its origin. Sampling the phase function for directions is crucial and should be done by importance sampling, as it heavily affects the final result, as depicted in Figure 6.5. To avoid unnecessary performance costs, we recommend pre-computing the sampling process and pass the results forward to the shader.

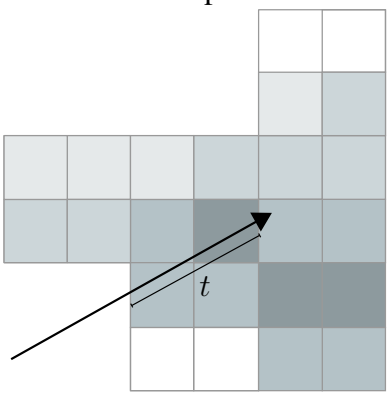
For each point along the sampling lobe oriented towards the ray direction we calculate the transmittance and incoming light. Transmittance is calculated by doing a sum of transmittances between the point inside the volume and each sample point along the lobe, then divide it by the number of sample points, thus returning the average, as presented in the first term of Equation 7. For the incoming light we do a sum for all light sources and, for each light source, another sum of the phase function between the sample point along the lobe and the light source position, multiplied by the transmittance between both. Then we multiply by the falloff term and divide by the number of sampled points along the lobe, hence giving an average of the incoming radiance for each given ray cast onto the volume as presented by the second term of Equation 7.

$$L(x,\omega) = L_b(x,\omega)\sum_{i=0}^{n}\frac{Tr(P_i)}{n} + \sum_{k=0}^{l}\sum_{i=0}^{n}\frac{f_p(P_i,L_k)Tr(P_i,L_k)\dfrac{L(P_i,L_k)}{||L_k-P_i||^2}}{n}. \quad (7)$$

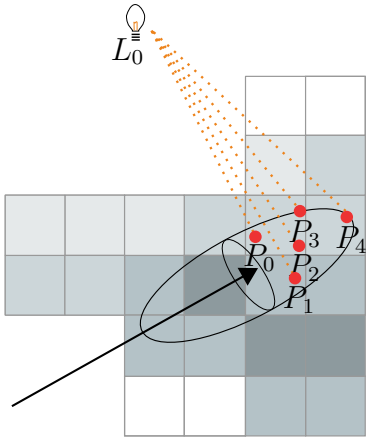
Figure 6.1: Lobe Estimator algorithm demonstrated in three steps.



(a) Ray cast at the volume border.



(b) Projected vector origin inside the volume using the mean path length multiplied by the depth factor, as depicted by t.



(c) Sample points on the lobe surface along the new vector origin. For each point, we calculate the transmittance and incoming light.

Source: The author.

To showcase our algorithm we used a scene with the Disney cloud at one quarter its original resolution with one light point right above it. In Figure 6.2 we compare the results of each algorithm side by side and in Table 6.1 we give the associated rendering times.

Note that the Monte Carlo rendering time is calculated as the total time it takes to converge to the number of samples per pixel selected. Here the Lobe Estimator approximates the Monte Carlo result quite well, partially softening some shadow areas, whereas the ray marching extrapolates the light intensity throughout the volume.

Table 6.1: Rendering times in milliseconds for each method. The lower the better. The Monte Carlo algorithm was rendered using 1000 SPP and 100 bounces. The Lobe Estimator algorithm used 100 steps for shadow rays and depth factor of 20. Ray Marching was used with 60 steps along the ray and 10 steps for each shadow ray.

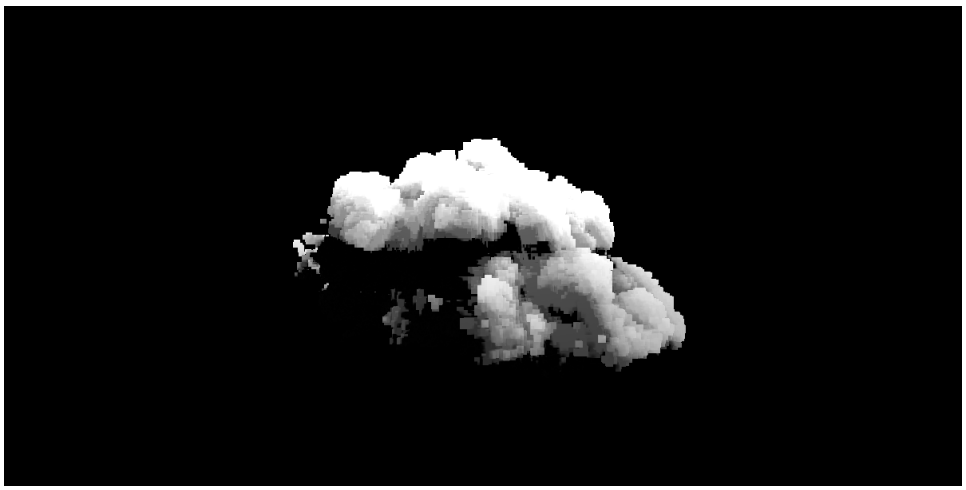
Method	**Time (ms)**
Lobe Estimator	55.5
Ray Marching	125
Monte Carlo	115236

As long as we have fewer points to sample compared to ray marching we should perform better. In Figures 6.3, 6.4, and 6.5 we show the impacts of each parameter of our algorithm. Note specially how the selection of points along the lobe impacts the final results, as depicted in Figure 6.5. Although not covered in this work, we strongly believe that, similarly as how importance sampling works, one must choose points that have major impact by selecting the ones with highest BSDF values, thus working well for static lights where we can pre-compute the highest BSDF values. It may be penalized in performance cost when considering dynamic lights, as one would need to calculate it in real-time.

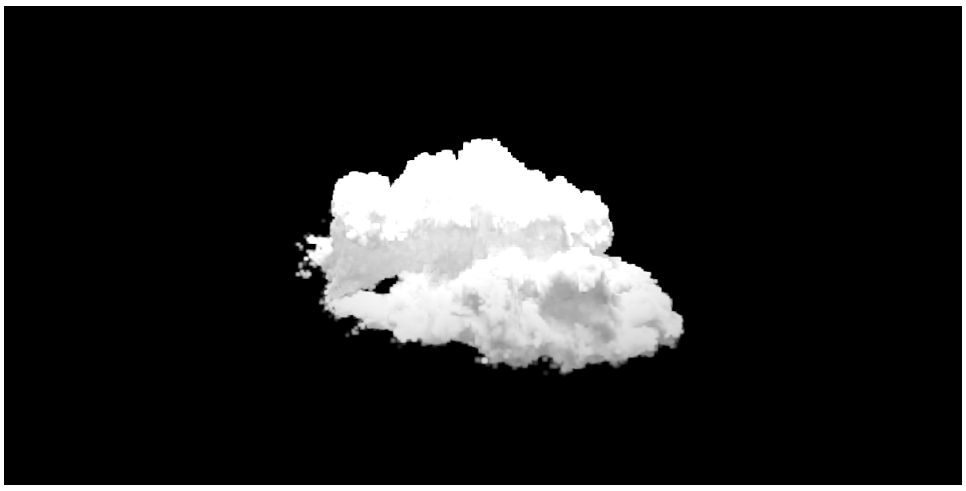
The main downside of our proposed algorithm are the need of fine tuning and it does not approximate well with low density participating media. However, on the upside it performs really well in real-time and is twice as fast as Ray Marching, thus being suitable for use cases where the application must be interactive.

Figure 6.2: Result for each algorithm under same scene configuration.

(a) Monte Carlo algorithm with 100 bounces and 1000 SPP.



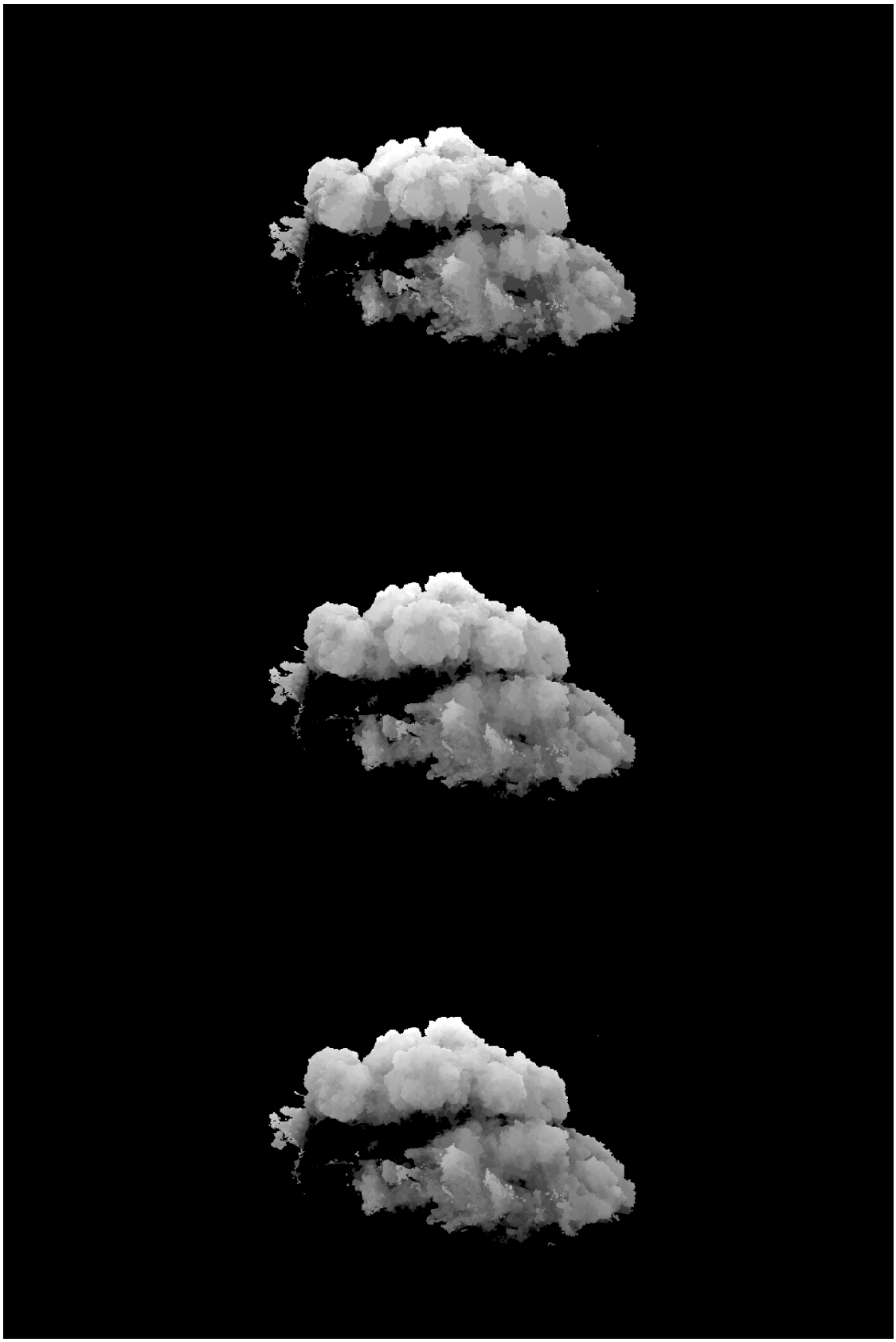
(b) Lobe Estimator algorithm rendered with depth parameter 20, 30 sampling points along the lobe, and 100 points towards the light.



(c) Ray Marching algorithm based on the implementation of (HILLAIRE, 2015a) with 60 steps along the ray and 10 shadow ray steps.

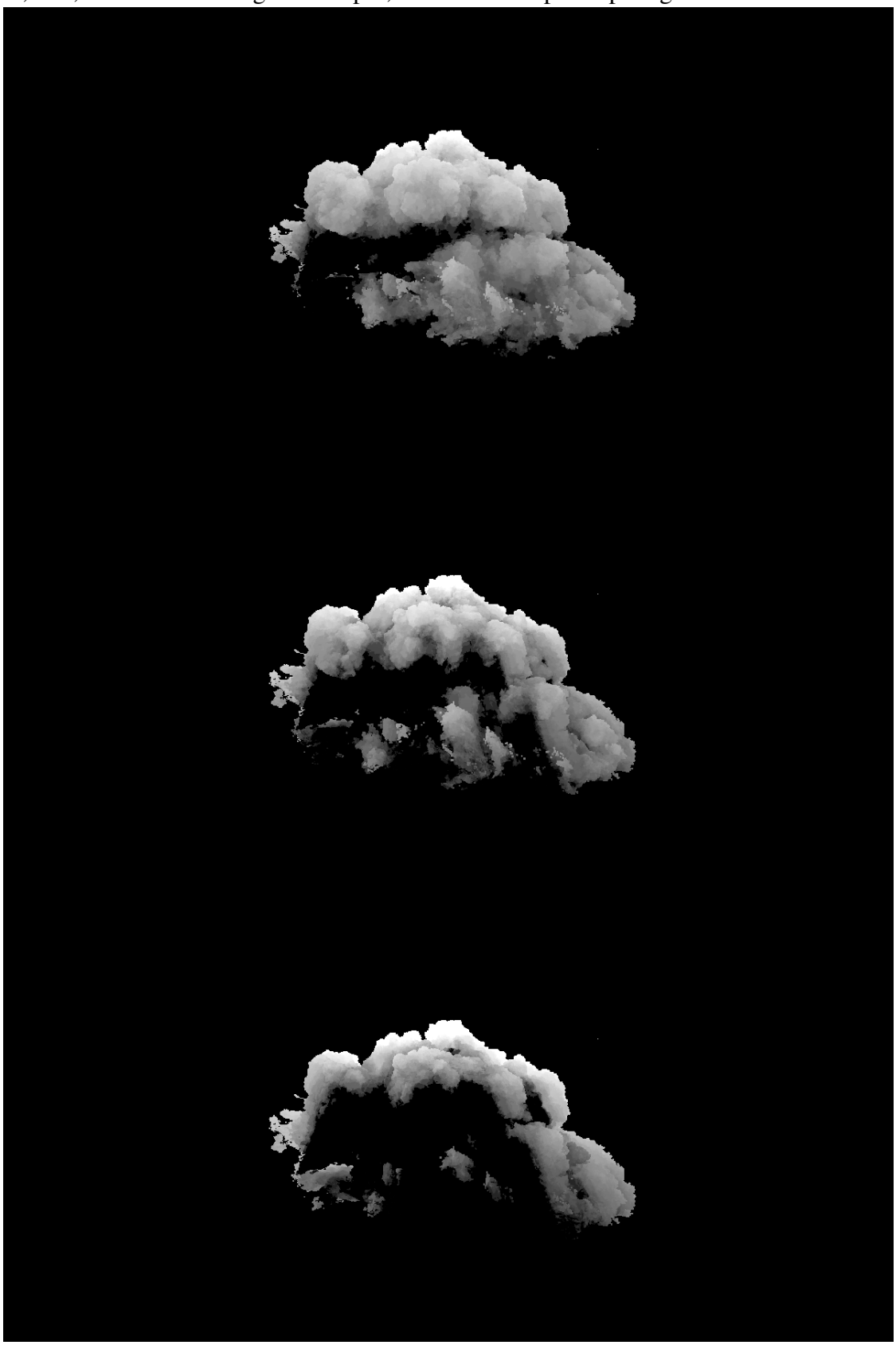
Source: The author.

Figure 6.3: Visual impact of changing the number of sampling points around the lobe. From top to bottom the number of sampled points is 10, 20, and 30. Note that the larger the number of sampled points the better the result, making the "washed out" appearance less noticeable.



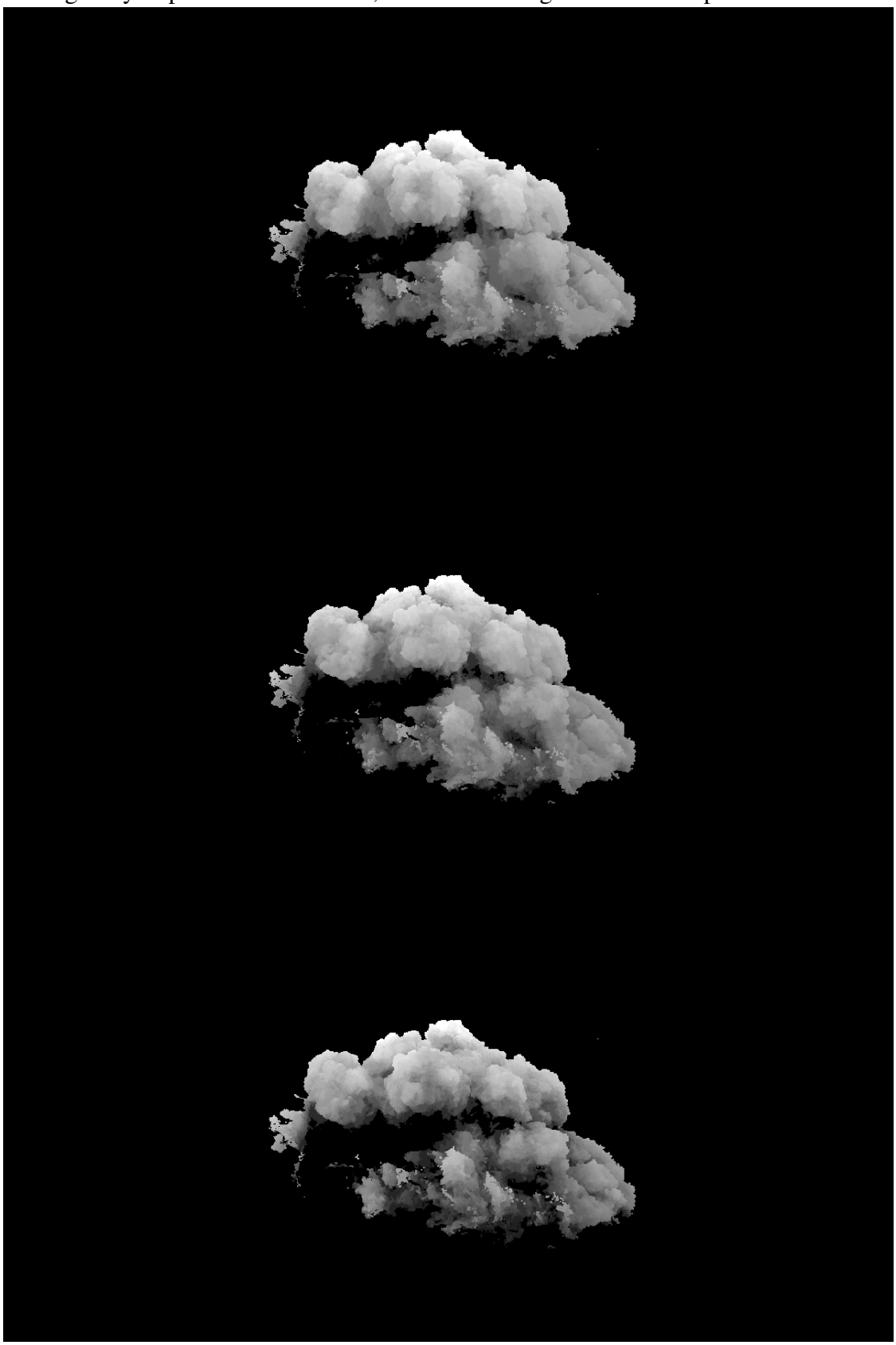
Source: The author.

Figure 6.4: Visual impact of changing the depth factor. From top to bottom the values are 20, 100, and 200. The larger the depth, the darker the participating medium.



Source: The author.

Figure 6.5: Visual impact of choosing different sets of sampling points along the lobe which greatly impacts the final result, hence indicating its extreme importance.



Source: The author.

7 CONCLUSION

To this day participating media are one of the most difficult and costly type of material to render. It demands large memory consumption, when not using procedural generation, and immense processing power due to its three-dimensional complexity. Its challenges also include dealing with low convergence rate, overlapping with another materials, and appropriate solutions for the Rendering Transfer Equation (RTE).

In the cinematographic industry, where the rendering process is usually comprised of rendering farms and time and memory constraints are more flexible, various algorithms are used and have been yielding good visual results. However, in the game industry, where the rendering pipeline is much more restrictive, usually having just a single GPU at the user's home, time and memory are a restriction many times more significant. In this type of environment various compromises must be made, and only in recent years the video game industry was able to incorporate proper volumetric rendering into their pipelines.

The two most common compromises made in a real-time scenario are related to rendering time, changing accurate methods for ones that are faster but introduce bias, and memory storage, where a single 3D texture file with resolution of 512^3 can cost up to 50 0MB. Considering we store only a single density floating-point value per voxel, 500 MB corresponds to 8.3% of the total space available in a GPU with 6 GB, the most common VRAM storage capacity nowadays, according to Steam August 2021 hardware and software survey (STEAM, 2021). Taking into account that a cloudscape has hundreds of clouds, it is completely impractical to expect such file sizes just for volumetric data.

We presented a perceptual analysis based on the JND CIELAB metric in order to verify how the human visual system perceives changes of rendering parameters and how we could save computation time by lowering their values without suffering from a huge visual impact. This study lays the foundation for better understanding of how the most accurate method, pah tracing, could be explored to improve its performance while maintaining image fidelity. Unfortunately, although we reached interesting performances, they are still not sufficient for real-time rendering. Hence, we use the visual cues obtained through our analysis to put together ideas on how we could approximate rendering for real-time.

After this analysis, we empirically proposed a new lobe sampling algorithm suitable for real-time, which, although not as good as path tracing, gives us suitable results for applications where the interactive aspect is of utmost importance.

For future work our lobe sampling algorithm could be improved by testing whether sampling multiple lobes at different consecutive depths does impact the final result positively at a moderate performance cost.

We believe in open science where society should benefit from its results and have free access to information. Hence, all shader code, implementation, and test scenes used throught this dissertation are publicly available at github.com/ibfernandes/NarvalEngine.

REFERENCES

BAUER, F. **Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution**. Siggraph - Advances in real-time rendering, 2019. Disponível em: <http://advances.realtimerendering.com/s2019/index.htm>.

BLINN, J. F. Models of light reflection for computer synthesized pictures. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 11, n. 2, p. 192–198, jul. 1977. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/965141.563893>.

BOUTHORS, A. et al. Interactive multiple anisotropic scattering in clouds. In: HAINES, E.; MCGUIRE, M. (Ed.). **I3D'08 - ACM Symposium on Interactive 3D Graphics and Games**. Redwood City, United States: ACM, 2008. (I3D '08 Proceedings of the 2008 symposium on Interactive 3D graphics and games), p. 173–182. Disponível em: <https://hal.inria.fr/inria-00333007>.

Chandrasekhar, S. **Radiative transfer.** [S.l.: s.n.], 1950.

DíAZ, J. et al. Real-time ambient occlusion and halos with summed area tables. **Computers and Graphics**, p. 337–350, 08 2010.

DISNEY. **Clouds data set**. 2021. Disponível em: <https://www.disneyanimation.com/data-sets/?drawer=/resources/clouds/>.

DORSEY, J.; RUSHMEIER, H.; SILLION, F. **Digital Modeling of Material Appearance**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. ISBN 9780080556710, 9780122211812.

ELEK, O. et al. Principal-ordinates propagation for real-time rendering of participating media. **Comput. Graph.**, Pergamon Press, Inc., Elmsford, NY, USA, v. 45, n. C, p. 28–39, dez. 2014. ISSN 0097-8493. Disponível em: <http://dx.doi.org/10.1016/j.cag.2014.08.003>.

ELLIS, S. R. et al. Generalizeability of latency detection in a variety of virtual environments. **Proceedings of the Human Factors and Ergonomics Society Annual Meeting**, v. 48, n. 23, p. 2632–2636, 2004. Disponível em: <https://doi.org/10.1177/154193120404802306>.

ENGEL, W. **GPU Pro 7 - Advanced Rendering Techniques**. 1. ed. [S.l.]: A K Peters/CRC Press, 2016. ISBN 9781498742535.

FONG, J. et al. **Production Volume Rendering**. 2017. Disponível em: <https://graphics.pixar.com/library/ProductionVolumeRendering/paper.pdf>.

FOUNDATION, A. S. **OpenVDB**. Disponível em: <https://www.openvdb.org/>.

GKIOULEKAS, I. et al. Understanding the role of phase function in translucent appearance. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 32, n. 5, p. 147:1–147:19, out. 2013. ISSN 0730-0301. Disponível em: <http://doi.acm.org.ezp-prod1.hul.harvard.edu/10.1145/2516971.2516972>.

HADWIGER et al. Gpu-accelerated deep shadow maps for direct volume rendering. **ACM SIGGRAPH/EG Conference on Graphics Hardware**, 09 2006.

HENYEY, L. C.; GREENSTEIN, J. L. Diffuse radiation in the galaxy. **The Astrophysical Journal**, v. 93, p. 70–83, 1941.

HERNELL, F.; LJUNG, P.; YNNERMAN, A. Interactive global light propagation in direct volume rendering using local piecewise integration. In: . [S.l.: s.n.], 2008. p. 105–112.

HERNELL, F.; LJUNG, P.; YNNERMAN, A. Local ambient occlusion in direct volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, v. 16, p. 548–559, 2010.

HILLAIRE, S. **Physically-based and Unified Volumetric Rendering**. 2015. Disponível em: <https://www.ea.com/frostbite/news/physically-based-unified-volumetric-rendering-in-frostbite>.

HILLAIRE, S. **Towards Unified and Physically-Based Volumetric Lighting in Frostbite**. Siggraph - Advances in real-time rendering, 2015. Disponível em: <http://advances.realtimerendering.com/s2015/index.html>.

HOFMANN, N. et al. Interactive path tracing and reconstruction of sparse volumes. **Proc. ACM Comput. Graph. Interact. Tech.**, Association for Computing Machinery, New York, NY, USA, v. 4, n. 1, abr. 2021. Disponível em: <https://doi.org/10.1145/3451256>.

JÖNSSON, D. et al. Historygrams: Enabling Interactive Global Illumination in Direct Volume Rendering using Photon Mapping. **IEEE Transactions on Visualization and Computer Graphics (TVCG)**, v. 18, n. 12, p. 2364–2371, 2012.

JÖNSSON, D. et al. A survey of volumetric illumination techniques for interactive volume rendering. **Comput. Graph. Forum**, v. 33, p. 27–51, 2014.

KNISS, J. et al. A model for volume lighting and modeling. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 9, n. 2, p. 150–162, abr. 2003. ISSN 1077-2626. Disponível em: <http://dx.doi.org/10.1109/TVCG.2003.1196003>.

KRAVCHENKO, B. et al. High-fidelity iridal light transport simulations at interactive rates. **Computer Animation and Virtual Worlds**, p. e1755, 01 2017.

KROES, T.; POST, F.; BOTHA, C. Exposure render: An interactive photo-realistic volume rendering framework. **PloS one**, v. 7, p. e38586, 07 2012.

KRONANDER, J. et al. Efficient visibility encoding for dynamic illumination in direct volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, v. 18, n. 3, p. 447–462, March 2012.

LAM, M. W. Y.; BARANOSKI, G. V. G. A predictive light transport model for the human iris. **Comput. Graph. Forum**, v. 25, p. 359–368, 2006.

LEVOY. Display of surfaces from volume data. **IEEE Computer Graphics and Applications**, v. 8, n. 3, p. 29–37, May 1988.

MAHY, M.; EYCKEN, L. V.; OOSTERLINCK, A. Evaluation of uniform color spaces developed after the adoption of cielab and cieluv. **Color Research & Application**, v. 19, n. 2, p. 105–121, 1994. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1520-6378.1994.tb00070.x>.

Max, N. Optical models for direct volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, v. 1, n. 2, p. 99–108, June 1995.

MILLER, B.; GEORGIEV, I.; JAROSZ, W. A null-scattering path integral formulation of light transport. **ACM Transactions on Graphics (Proceedings of SIGGRAPH)**, v. 38, n. 4, jul. 2019. ISSN 0730-0301.

NOVáK, J.; SELLE, A.; JAROSZ, W. Residual ratio tracking for estimating attenuation in participating media. **ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)**, v. 33, n. 6, nov. 2014.

PEGORARO, V. **Handbook of Digital Image Synthesis**. 1. ed. [S.l.]: A K Peters/CRC Press, 2016. ISBN 9781315395227.

PHARR, M.; JAKOB, W.; HUMPHREYS, G. **Physically Based Rendering: From Theory to Implementation (3rd ed.)**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. 1266 p. ISBN 9780128006450. Disponível em: <https://www.pbrt.org/>.

RAAB, M.; SEIBERT, D.; KELLER, A. Unbiased global illumination with participating media. In: KELLER, A.; HEINRICH, S.; NIEDERREITER, H. (Ed.). **Monte Carlo and Quasi-Monte Carlo Methods 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 591–605. ISBN 978-3-540-74496-2.

ROPINSKI, T.; DORING, C.; REZK-SALAMA, C. Advanced volume illumination with unconstrained light source positioning. **IEEE Computer Graphics and Applications**, v. 30, n. 6, p. 29–41, Nov 2010.

ROPINSKI, T. et al. Interactive volume rendering with dynamic ambient occlusion and color bleeding. **Comput. Graph. Forum**, v. 27, p. 567–576, 04 2008.

SCHLEGEL, P.; MAKHINYA, M.; PAJAROLA, R. Extinction-based shading and illumination in gpu volume ray-casting. **IEEE Transactions on Visualization and Computer Graphics**, v. 17, n. 12, p. 1795–1802, Dec 2011.

SCHOTT, M. et al. A directional occlusion shading model for interactive direct volume rendering. **Comput. Graph. Forum**, v. 28, p. 855–862, 2009.

SHARMA, G. **Digital Color Imaging handbook**. 1. ed. [S.l.]: CRC Press LLC, 2003. ISBN 0-8493-0900-X.

SHNEIDER, A. **The Real-time Volumetric Cloudscapes of Horizon: Zero Dawn**. Siggraph - Advances in real-time rendering, 2015. Disponível em: <http://advances.realtimerendering.com/s2015/index.html>.

SHNEIDER, A. **Nubis: Authoring Real-Time Volumetric Cloudscapes with the Decima Engine**. Siggraph - Advances in real-time rendering, 2017. Disponível em: <http://advances.realtimerendering.com/s2017/index.html>.

SOLTÉSZOVÁ, V. et al. A multidirectional occlusion shading model for direct volume rendering. **Comput. Graph. Forum**, v. 29, p. 883–891, 2010.

STEAM. **Steam Hardware and Software Survey**. 2021. Disponível em: <https://store.steampowered.com/hwsurvey?platform=combined>.

SUNDEN, E.; YNNERMAN, A.; ROPINSKI, T. Image plane sweep volume illumination. **IEEE Transactions on Visualization and Computer Graphics**, v. 17, n. 12, p. 2125–2134, Dec 2011.

ZHANG, C.; XUE, D.; CRAWFIS, R. Light propagation for mixed polygonal and volumetric data. In: **International 2005 Computer Graphics**. [S.l.: s.n.], 2005. p. 249–256.

ZHANG, Y.; MA, K.-L. Fast global illumination for interactive volume visualization. In: **Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games**. New York, NY, USA: ACM, 2013. (I3D '13), p. 55–62. ISBN 978-1-4503-1956-0. Disponível em: <http://doi.acm.org/10.1145/2448196.2448205>.

APPENDIX A — UMA APROXIMAÇÃO PARA DISPERSÃO MÚLTIPLA EM ILUMINAÇÃO VOLUMÉTRICA VOLTADA PARA RENDERIZAÇÃO EM TEMPO REAL

Renderização volumétrica é uma área da computação gráfica voltada para técnicas responsáveis pelo cálculo e resolução da equação de renderização em mídias participativas, isto é, volumes compostos de partículas esparsas tais como poeira, gotículas de água, nuvens e fumaça. Muitas técnicas de iluminação volumétrica foram desenvolvidas ao longo do tempo para solucionar a equação de renderização. Entretanto, ainda existem muitas limitações quando trabalhando com *multiple scattering* em tempo real devido à sua complexidade e escala.

Na comunidade de computação gráfica, a constante busca por sintetizadores de imagens realistas em tempo real foi um dos seus maiores e mais ambiciosos objetivos desde o início. Nos anos mais recentes houve um esforço para tentar trazer a propriedade de *multiple scattering* para tempo real, com limitações nos tamanhos de grid e capacidade de processamento. Muito embora seja possível renderizar o efeito de *multiple scattering* em tempo real com restrições, ainda é um grande desafio a ser superado devido sua complexidade. Já é muito custoso e complexo calcular e traçar raios dentro de uma única mídia participativa, mas as coisas tornam-se especialmente complicadas quando se leva em conta múltiplas mídias e suas interações com outros objetos na cena.

Estas técnicas de renderização volumétrica partem da premissa de que tais volumes são compostos por partículas tão pequenas e esparsas que o modelo de renderização é melhor modelado como um processo probabilístico. Algumas técnicas são discutidas no Capítulo 4. Entretanto, esse mesmo processo probabilístico sofre de ruídos e baixa convergência, muito embora seja capaz de produzir resultados fisicamente corretos. Para calcular a interação da luz com um volume deve-se levar em conta três propriedades: absorção, emissão e dispersão. Esse trabalho propõe uma nova solução em tempo real para aproximar o efeito produzido por *multiple scattering*, comumente usado em renderers offline.

Nossa abordagem foi explorar o sistema de percepção visual humano para acelerar o processamento. Dado duas imagens, nós usamos a métrica do CIE que diz que duas imagens serão percebidas de maneira similar perante o olho humano se a distância Euclidiana no espaço de cores do CIELAB for menor que 2.3. Desta forma, nós usamos esta premissa para guiar nossas investigações no Capítulo 5. Nossos resultados mostram

que podemos reduzir de 10^5 para 10^4 amostras por pixel de maneira que a diferença para o olho humano seja negligenciável, permitindo reduzir o tempo de renderização em até 10 vezes toda vez que reduzimos a quantidade de amostras por pixel em 10. De maneira similar, é possível reduzir o número de *bounces* de 1000 para 100 de maneira negligenciável ao olho humano, permitindo reduzir o tempo de renderização em quase pela metade. Os principais resultados das análises são demonstrados nas Figuras 5.4, 5.6, 5.8 e 5.9.

No Capítulo 6 subsequente é proposto um novo algoritmo em tempo real, entitulado Lobe Estimator, ou "Estimador em Lobo", em tradução literal, que aproxima esses comportamentos e parametros discutidos enquanto mantém uma performance até duas vezes mais rápida que o clássico *Ray Marching* utilizado em tempo real.