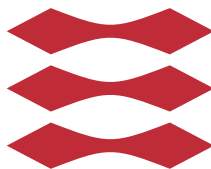


# Scattering of Photon Differentials in Realistic Rendering of Volumes

Andrea Luongo

DTU



Kongens Lyngby 2015

Technical University of Denmark  
Department of Applied Mathematics and Computer Science  
Richard Petersens Plads, building 324,  
2800 Kongens Lyngby, Denmark  
Phone +45 4525 3031  
[compute@compute.dtu.dk](mailto:compute@compute.dtu.dk)  
[www.compute.dtu.dk](http://www.compute.dtu.dk)

"Don't Panic."

Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

**d**

---

# Summary (English)

---

When light travels through a participating medium, like fog or water, there are phenomena, as absorption or scattering, which affect the propagation of the light and give rise to interesting effects, e.g. atmospheric haze or volume caustic.

Volumetric Photon Mapping is an efficient method for rendering participating media. However, as it is based on density estimation it suffers from the same trade-off between noise and bias as traditional photon mapping.

By using photon differentials it is possible to improve this trade-off, but there are still some unsolved problems regarding the use of this technique for rendering volumes.

The goal of this thesis is to investigate ways of improving existing algorithms for rendering volumes by using photon differentials.

A new estimate of the radiative transfer equation based on photon differentials will be presented, together with a description of how to implement an efficient GPU technique for rendering participating media by using the *NVIDIA*® *OptiX*<sup>TM</sup> ray tracing engine.



# Preface

---

This thesis was prepared at the DTU Compute department in fulfilment of the requirements for acquiring a M.Sc. in Digital Media Engineering.

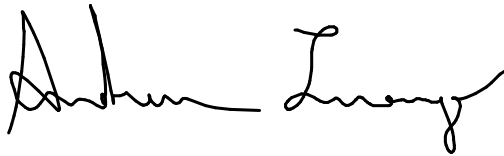
The thesis deals with the rendering of participating media using a GPU algorithm based on photon differentials. Participating media includes all the media, like fog, water, smoke and many others, which affect the light that passes through them by scattering it or absorbing it.

The author's interest for Computer Graphics and Physically Based Rendering has grown during the M.Sc. in Digital Media Engineering, where he followed courses from both the Computer Graphics and the Computer Games study lines. Professor Jeppe Revall Frisvad of DTU Compute proposed a research-oriented M.Sc. thesis which goal was to investigate and improve existing algorithms for rendering volumes using photon differentials. The topic represented an excellent opportunity to research in Physically Based Rendering techniques, and so the author registered his application for this master thesis, *Scattering of Photon Differentials in Realistic Rendering of Volumes*.

The thesis consists of a GPU software implementation done by using the *NVIDIA*<sup>®</sup> *OptiX*<sup>™</sup> ray tracing engine [PBD<sup>+</sup>10], based on the *NVIDIA*<sup>®</sup> *CUDA*<sup>®</sup> GPU computing architecture, and this report. The initial OptiX framework was taken from DTU course 02756, *Physically Based Rendering*, and then expanded in or-

der fit the needs of the thesis. All the screenshots in this report are generated using software developed by the author. All the other images, unless a reference is reported in the caption, were created by the author.

Lyngby, 05-June-2015

A handwritten signature in black ink, appearing to read 'Andrea Luongo'. The signature is written in a cursive style with a large initial 'A' and 'L'.

Andrea Luongo



# Acknowledgements

---

I would like to thank my supervisor for this MSc project, Jeppe Revall Frisvad, for the help and support he provided during the whole duration of this thesis. The long meetings in his office have been very helpful in the development of this project.

I would also like to thank the University of Padua for the opportunity to participate in the T.I.M.E double degree program and to complete a Master Degree abroad. Special thanks are due to professor Maria Elena Valcher, for helping me with all the bureaucracy the T.I.M.E. project requires, and to professor Pietro Zanuttigh, for accepting to be my supervisor for my MSc defense in Italy.

I would like to thank all the wonderful friends that I have met during these two years at DTU for all the special moments spent together, my Italian friends *Sfizzuppoli* and *Piscini* for the daily chats that have helped me not to miss home, and especially thanks to Cippo for helping me with my English, and to Sebastiano, "The Man Above the Math", for the mathematical help. Special thanks to Marta for the support she gave me in my difficult times, and for being patient with me during these two long years. Finally I would like to thank my brother Fabio, my grandparents Lina and Luigi, my father Sabatino, and my mother Lorena for everything they have done and all the sacrifices they made to help me reach this goal.



# Contents

---

<b>Summary (English)</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem Statement . . . . .	3
1.3 Thesis Outline . . . . .	4
<b>2 Noise and Bias in Photon Mapping</b>	<b>7</b>
2.1 Photon Mapping . . . . .	7
2.2 Noise and Bias . . . . .	10
2.3 Photon Differentials . . . . .	12
<b>3 Volume Rendering</b>	<b>17</b>
3.1 Light Transport in a Participating Medium . . . . .	17
3.1.1 Emission . . . . .	18
3.1.2 Absorption . . . . .	18
3.1.3 Scattering . . . . .	19
3.1.4 Radiative Transfer Equation . . . . .	21
3.2 Related Work . . . . .	22
3.2.1 Volumetric Photon Mapping . . . . .	22
3.2.2 The Beam Radiance Estimate . . . . .	24
3.2.3 Photon Differentials . . . . .	26
3.2.4 Photon Beams . . . . .	27

---

<b>4</b>	<b>Method</b>	<b>29</b>
4.1	Photon Pass . . . . .	29
4.1.1	Photon Differentials for Volumes . . . . .	30
4.1.2	Sampling Techniques in Participating Media . . . . .	32
4.2	Ray Tracing Pass . . . . .	35
<b>5</b>	<b>Implementation</b>	<b>41</b>
5.1	Photon Differential Splatting for Surfaces . . . . .	42
5.2	Volume Rendering of Participating Media Using Photon Differentials . . . . .	48
5.2.1	Volume Rendering based on Eye Map . . . . .	49
5.2.2	Volume Rendering based on Photon Map . . . . .	55
5.3	Research and Improvement . . . . .	61
<b>6</b>	<b>Results</b>	<b>63</b>
6.1	Absorption and Scattering . . . . .	63
6.2	Isotropic Kernel vs Anisotropic Kernel . . . . .	68
6.3	Analysis of the Radiance Estimate . . . . .	71
<b>7</b>	<b>Conclusions and Future Work</b>	<b>79</b>
<b>A</b>	<b>Appendix A</b>	<b>83</b>
	<b>Bibliography</b>	<b>87</b>

# Introduction

---

The use of Computer Graphics is increased very fast in the years and it is required in a great variety of fields, like films, video games, television but also in medical equipment and almost every simulation software and many other applications. This fast growth has entailed the need to develop techniques and algorithms more and more accurate in order to get results as close as possible to the reality.

An interesting and useful phenomenon to simulate is the behaviour of light in the presence of a participating medium like fog or smoke. By simulating this effect it is possible to render scenes with atmospheric haze, crepuscular rays, fog, smoke, water and many other effects. In Figure 1.1a and Figure 1.1b two of these effects are shown.

When the light enters a medium it may be absorbed or scattered, and part of the scattered light may reach the eye of the viewer and for this reason the light is visible even when there are no surfaces to reflect it.

## 1.1 Background

The scattering of light in a participating medium is a complex phenomenon and in order to render it it is necessary to solve the radiative transfer equation, RTE, [Cha60] combined with the rendering equation [Kaj86].



(a) Foggy environment [Fur14].



(b) Crepuscular Rays [Lac06].

**Figure 1.1:** Two examples of light travelling through a participating medium.

An accurate solution to these equations is costly to compute and different techniques were developed in order to overcome this problem. Some of these techniques are based on stochastic sampling and Monte Carlo integration, like bidirectional path tracing ([LW93], [LW96] and [VG94]) or Metropolis light transport ([MP00]). These approaches are guaranteed to converge to the right solution but the convergence is slow, in particular it requires a long time to get a noise-free result when the scene contains both participating media and specular surfaces, situation that is very common in physical scenes.

An approach that is not affected by these problems is Volumetric Photon Mapping [JC98]. This method is based on density estimation and it can robustly

handle specular materials, it is also less affected by noise but it introduces bias and requires more memory for the computation.

Jarosz et al. [JZJ08] propose a new gathering technique called the "beam radiance estimate" to improve the efficiency of Volumetric Photon Mapping. Schjøth [Sch09] extends the concept of photon differentials [SFES07] to include participating media. Jarosz et al. [JNSJ11] and [JNT<sup>+</sup>11] propose a method to reduce noise and bias by using photon beams.

## 1.2 Problem Statement

Volume Photon Mapping [JC98] is an extension of the Photon Mapping method [JC95] [Jen96] and it is able to render isotropic, anisotropic, homogeneous and heterogeneous media and it also supports specular interactions with materials. This technique makes use of density estimation and, as for Photon Mapping, it introduces bias in order to reduce noise. It also uses ray marching in order to gather photons at interval along the ray, and this process introduces a trade-off problem between estimation accuracy and computation time.

The "beam radiance estimate" proposed by Jarosz et al. [JZJ08] avoid the use of ray marching by performing a single query to gather all the data that are needed to perform the density estimation. In order to reduce bias they also propose to adaptively change the bandwidth of the density estimation by performing a pilot estimate of the local density.

Photon differentials [SFES07] were introduced to improve the trade-off problem of Photon Mapping so that with the same number of photons both the noise, and the bias were reduced. As this technique has proved to work very well for rendering caustics on surfaces, Schjøth [Sch09] proposes to extend it to volumes in order to compute variable bandwidths for the density estimation instead of performing the expensive pilot estimate proposed by Jarosz et al. [JZJ08]. The approach introduced by Schjøth to scatter photon differentials is based on heuristics and it can be improved.

Jarosz et al. [JNSJ11] point out that by using a particle representation to store light paths some of the information accumulated during photon tracing are lost. They propose to store *photon beams* instead of photons in order to have a more accurate estimate, and they also use photon differentials to adapt the shape and the size of the photon beams.

In a later work, Jarosz et al. [JNT<sup>+</sup>11] present a progressive radiance estimate based on photon beams and they also present different implementations of the algorithm, one of which is implemented on the GPU using the *NVIDIA*® *OptiX<sup>TM</sup>* ray tracing engine [PBD<sup>+</sup>10].

The goals of this thesis project are to investigate and improve existing al-

gorithms for rendering volumes using photon differentials. The implemented method uses as starting point the beam radiance estimate [JZJ08] and the work done by Schjøth [Sch09], but a more precise estimate of the radiative transfer equation is developed. Furthermore, in order to improve the performance and to provide a progressive technique that incrementally updates the results, the algorithm has been implemented on the GPU by using the *NVIDIA® OptiX™* ray tracing engine [PBD<sup>+</sup>10], taking inspiration from the work done by Jarosz et al. [JNT<sup>+</sup>11].

The work presented in this thesis provides a solid starting point from which it is possible to continue researching on photon differentials scattering.

### 1.3 Thesis Outline

In this chapter the topic of this thesis project has been presented, some of the existing algorithms for rendering volume have been introduced, and a description of the objectives of this project has been made.

In Chapter 2, a brief description of the Photon Mapping algorithm for global illumination and its problems will be given; it will also be presented a technique to reduce noise and bias based on photon differentials. The goal of Chapter 2 is to help the reader to understand what are the typical issues that the methods based on density estimation introduce.

In Chapter 3 participating media are introduced. In the first section the phenomena that affect the light travelling through a participating medium are described, and it is shown how it is possible to derive the radiative transfer equation (RTE) which accounts for all this phenomena. In the end, a more detailed description than the one provided in Chapter 1 of the estimate techniques for rendering volumes that have been developed over the years will be presented.

In Chapter 4, the theory and the methods used in the implementation of this project will be explained. It will be shown how a more accurate estimate of the RTE than the one presented by Schjøth [Sch09] can be computed.

In Chapter 5, the details of how the algorithm has been implemented on the GPU are provided. In the first section it is presented the first step of the implementation which is a GPU version of the photon differential splatting technique [FSES14] for surfaces. In the second section are presented the two different approaches that have been tried in order to implement volume rendering. In the third and last section the estimate of the RTE found in Chapter 4 has been implemented on the GPU.

In Chapter 6, the results are shown and it will be presented a comparison between the estimate used by Schjøth and the estimate developed in this thesis. It will also shown how the use of photon differentials and anisotropic filtering improves the quality of the results compared to isotropic filtering as used in *the*



*beam radiance estimate* or in classical Volumetric Photon Mapping. Finally, in Chapter 7 the conclusions and some possible extensions to the method will be given.



## CHAPTER 2

# Noise and Bias in Photon Mapping

---

In this chapter a description of Photon Mapping and its problems are presented. The goal is to show the trade-off problem between noise and bias arising from the use of density estimation in Photon Mapping. This content will be an essential introduction for the next chapters.

First a brief description of the Photon Mapping global illumination method for surfaces will be presented. Then it will be presented an analysis of the noise-bias trade-off. Finally, the chapter will be concluded with an introduction to Photon Differentials, a technique to decrease both noise and bias.

## 2.1 Photon Mapping

The simulation of global illumination in a general environment requires solving the *rendering equation* or *light transport equation*, LTE, introduced by Kajiya

[Kaj86], for each point of the scene:

$$\begin{aligned} L_o(\mathbf{p}, \vec{\omega}_o) &= L_e(\mathbf{p}, \vec{\omega}_o) + \int_{4\pi} f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_i) L_i(\mathbf{p}, \vec{\omega}_i) |\cos \theta_i| d\vec{\omega}_i \\ &= L_e(\mathbf{p}, \vec{\omega}_o) + L_r(\mathbf{p}, \vec{\omega}_o). \end{aligned} \quad (2.1)$$

This equation describes the exitant radiance  $L_o(\mathbf{p}, \vec{\omega}_o)$  in direction  $\vec{\omega}_o$  from a point  $\mathbf{p}$  on a surface in terms of the emitted radiance  $L_e(\mathbf{p}, \vec{\omega}_o)$ , the bidirectional scattering distribution function (BSDF)  $f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_i)$ , and the incident radiance  $L_i(\mathbf{p}, \vec{\omega}_i)$  coming from all the possible directions  $\vec{\omega}_i$ .

There is a problem related to equation (2.1): it does not admit a general solution and, since the complexity of a scene making use of physically based BSDF models and arbitrary geometry is very high, the only feasible approach is to use a numerical solution technique.

One way to proceed is by making use of stochastic sampling and Monte Carlo integration (e.g. path tracing). This approach guarantees a convergence to the exact solution to (2.1), but it has several drawbacks. First of all, it is time-consuming and it is affected by noise. Second, it can be used to simulate both specular and diffuse materials, but it has problems when it comes to render light paths of the type  $LS^+DE^1$ , Figure 2.1.

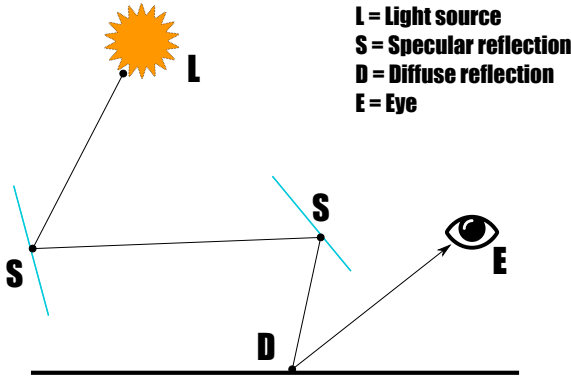


Figure 2.1: Example of  $LS^+DE$  light path.

This kind of path gives rise to caustics and they are very challenging to render

<sup>1</sup>Light Transport notation [Hec90]. In this case it denotes a light path starting from a light source and going through one or more specular reflections or refractions before reaching a diffuse surface and then the eye.

with path tracing, and the results are usually affected by high frequency noise because the light often follows a low-probability path.



**Figure 2.2:** Caustic generated by a glass of water [Ott06].

Photon Mapping was presented by [JC95] and [Jen96] as a method capable of simulating global illumination without any restrictions, i.e. it supports all kind of objects and materials and it can simulate caustics and indirect illumination efficiently. The Photon Mapping algorithm is divided in two passes: photon maps creation and final rendering.

- **Photon Maps creation:** In the first pass packets of energy, called *photons*, are emitted from the light sources and they are traced through the scene using a method similar to path tracing.

Two different data structures are created, a *caustic photon map* and a *global photon map*, and every time a photon hits a diffuse surface it is stored in the appropriate photon map: if it hits one or more specular surfaces before reaching a diffuse surface, then the photon and its information are stored in the *caustic photon map* otherwise they are stored in the *global photon map*. These two maps represent how the light is distributed in the scene; the accuracy of this representation depends on the number of photons that are emitted from the light sources and from the sizes of the photon maps.

The data structure behind a photon map is a multidimensional search tree, *kd-tree* [Ben75], and this is a good choice since many range searches are required in the next step and they are proved to be very efficient with a kd-tree.

- **Rendering:** In the second pass Monte Carlo ray tracing is used. Rays are traced from the eye through each pixel and, if they hit a point on a surface, the exitant radiance,  $L_o(\mathbf{p}, \vec{\omega}_o)$ , from that point has to be computed using (2.1). While the emitted radiance,  $L_e(\mathbf{p}, \vec{\omega}_o)$ , can be computed directly

from the surface definition, the reflected radiance,  $L_r(\mathbf{p}, \vec{\omega}_o)$ , needs some more computation.

The reflected radiance can be splitted in four different contributions: the radiance coming directly from the light sources, the radiance coming from specular surfaces, the radiance coming from multiple diffuse reflections and the radiance coming from  $LS^+DE$  paths (caustic). The first two contributions can be easily computed, while the estimate of the last two requires the informations stored in the photon maps.

An estimate of the radiance  $L_r(\mathbf{p}, \vec{\omega}_o)$  leaving a point  $\mathbf{p}$  along a direction  $\vec{\omega}_o$  can be done by searching in the photon maps the  $N$  photons closest to  $\mathbf{p}$  and since each photon  $j$  represents flux,  $\Delta\Phi_j$ , arriving at  $\mathbf{p}$  from a direction  $\vec{\omega}_i$  and, by keeping in mind the definition of radiance  $L = \frac{d^2\Phi}{dAd\vec{\omega}}$ ,  $L_r(\mathbf{p}, \vec{\omega}_o)$  is given by:

$$\begin{aligned} L_r(\mathbf{p}, \vec{\omega}_o) &= \int_{4\pi} f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_i) \frac{d^2\Phi_i(\mathbf{p}, \vec{\omega}_i)}{dAd\vec{\omega}_i} d\vec{\omega}_i \\ &\approx \sum_{j=1}^N f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_{i,j}) \frac{\Delta\Phi_j(\mathbf{p}, \vec{\omega}_{i,j})}{\pi r^2}, \end{aligned} \quad (2.2)$$

where  $r$  represents the radius of the smallest sphere centred at  $\mathbf{p}$  and enclosing all the  $N$  photons, this estimate is also known as *k'th nearest neighbour kernel estimate*. The radius  $r$  will be denoted as *bandwidth* of the estimate.

An alternative approach proposed by Jensen [Jen96] is to choose a fixed value for the bandwidth  $r$  and then use all the photons inside the sphere of radius  $r$ .

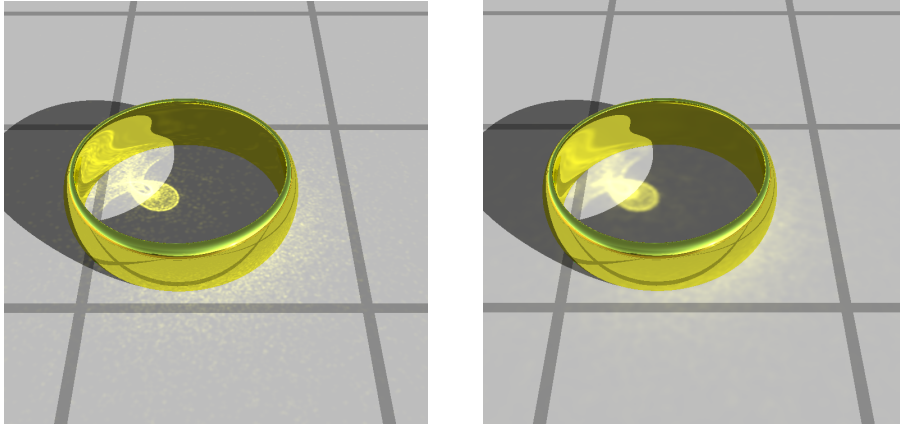
## 2.2 Noise and Bias

In the first pass of Photon Mapping, photons are stochastically sampled and traced from the light sources through the scene. This sampling process introduces noise that in principle could be removed by increasing the number of photons stored in the photon maps, but unfortunately this will require a lot of memory and computation time.

Thanks to the density estimate in equation (2.2), the noise can be reduced. Unfortunately, that introduces some systematic error, *bias*, that most of the time is visible as a blurring of the illumination.

For example, if  $N$  is large and the density of photons is low, then the final result will be blurry because the value of  $\Delta A = r^2\pi$  will be large. In situations where the illumination changes slowly from one point to another (e.g. diffuse surfaces and indirect illumination) the blurriness is not a big problem, but when

the change is quick (e.g. caustics and shadows) and the illumination features should be sharp, the blurriness becomes a real problem and many details could be lost. On the other hand if the value of  $N$  is small more details are preserved but the overall result is noisier.



(a) Photon Mapping with bandwidth  $r = 0.5$ .

(b) Photon Mapping with bandwidth  $r = 5$ .

**Figure 2.3:** The two images are rendered with the same number of photons but different bandwidths.

In Figure 2.3 two renderings of the same scene are shown: a gold ring generates a cardioid caustic with light coming from a directional light source. The images contain a total of  $10^6$  photons and a constant bandwidth approach is used. In Figure 2.3a it is visible how, in this scene, a bandwidth  $r = 0.5$  is not large enough to remove the noise but it provides a caustic with sharp edges, while the bandwidth  $r = 5$  in Figure 2.3b is too large and the density estimate removes part of the noise, but it introduces blurring.

Different techniques have been proposed to reduce bias. It is possible to introduce a filter to weigh the different samples in relation with their distance from the point  $\mathbf{p}$ , and (2.2) can be reformulate as:

$$L_r(\mathbf{p}, \vec{\omega}_o) = \frac{1}{r^2} \sum_{j=1}^N f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_{i,j}) \Delta\Phi_j(\mathbf{p}, \vec{\omega}_{i,j}) K\left(\frac{\|\mathbf{p} - \mathbf{x}_j\|}{r}\right), \quad (2.3)$$

where  $\mathbf{x}_j$  represents the position of photon  $j$  and  $K(y)$  is a symmetric and normalized function known as *kernel function*. For example Jensen [Jen96] uses a cone filter  $K(y) = \max(0, 1 - |y|) \frac{3}{2\pi}$ .

Jensen [JC95] proposes a method called *differential checking* that makes sure that the photons used in the estimate do not belong to distinct illumination features. Myszkowski [Mys97] proposes a method similar to the differential checking of [JC95], but more robust and easier to control, while Schregle [Sch03] improve the performance of [Mys97]. A technique to reduce bias by using statistics is presented by Walter [Wal98].

An interesting approach is the one proposed by Schjøth [SFES07] based on photon differentials and it will be shown in the next section.

## 2.3 Photon Differentials

The idea behind photon differentials is to exploit the coherence of one photon with its neighbours: when a photon is traced, its neighbours will tend to follow almost exactly the same path and this can be simulated by tracing an imaginary bundle of particles along with the photon. In this way a photon is not seen anymore as a single particle, but as a beam of light that changes shape according to its interactions along the path. This beam of light is represented by using ray differentials, as proposed by Igehy [Ige99].

When a photon is emitted, a position  $\mathbf{x}(u, v)$  on the light source and a direction  $\vec{\omega}(\theta, \phi)$  are sampled<sup>2</sup>. The ray representing the photon is given by

$$\mathbf{r}(u, v, \theta, \phi, t) = \mathbf{x}(u, v) + t\vec{\omega}(\theta, \phi). \quad (2.4)$$

The beam of light containing the photon and its neighbours can be approximated by considering the first order derivatives of  $\mathbf{x}(u, v)$  and  $\vec{\omega}(\theta, \phi)$  and by building two pairs of differential vectors:

$$D\mathbf{x} = \left\langle \frac{\partial \mathbf{x}}{\partial u}, \frac{\partial \mathbf{x}}{\partial v} \right\rangle, \quad (2.5)$$

$$D\vec{\omega} = \left\langle \frac{\partial \vec{\omega}}{\partial \theta}, \frac{\partial \vec{\omega}}{\partial \phi} \right\rangle, \quad (2.6)$$

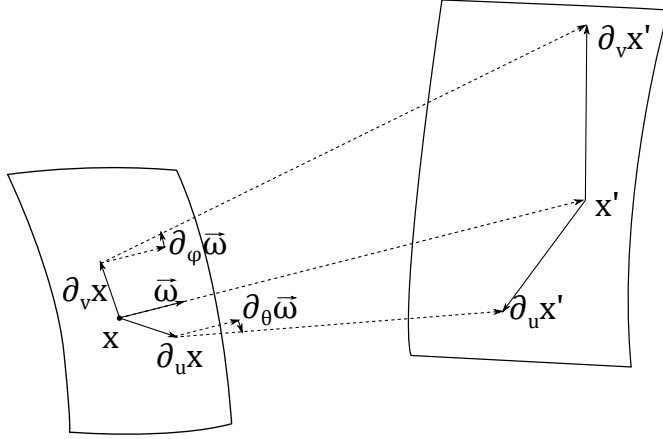
where (2.5) are called positional differential vectors and (2.6) directional differential vectors.

The differential vectors are traced through the scene along with the photon (Figure 2.4) and, at every intersection with a surface they are updated as in [Ige99]. The projections of the positional differential vectors onto a surface tangent to the intersected object define a parallelogram called *ray footprint*, Figure 2.5.

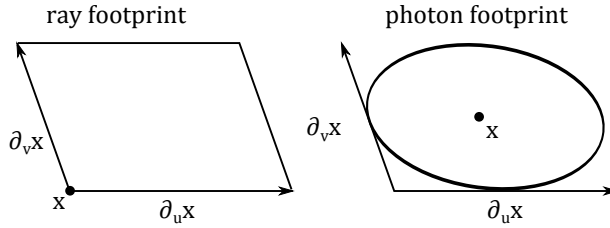
Schjøth [SFES07] and Frisvad [FSES14] define as *photon footprint* the ellipse with the maximum area inscribed in the ray footprint: the semi-axes correspond to  $\frac{1}{2}D\mathbf{x}$  and the center is the photon position  $\mathbf{x}$ .

<sup>2</sup> $u$  and  $v$  are a parametrization of the light source surface and  $\theta$  and  $\phi$  are a parametrization of the emission solid angle.





**Figure 2.4:** Propagation of photon differentials from  $\mathbf{x}$  to  $\mathbf{x}'$ .



**Figure 2.5:** Ray footprint and photon footprint.

The area of the photon footprint is given by:

$$A = \frac{\pi}{4} \left| \frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v} \right|. \quad (2.7)$$

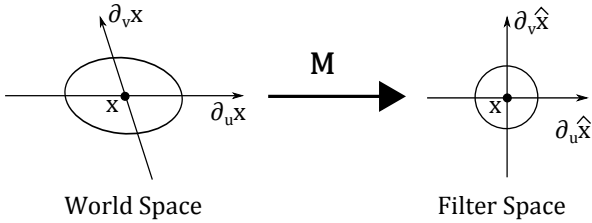
The radiance estimate at a point  $\mathbf{p}$  becomes:

$$L_r(\mathbf{p}, \vec{\omega}_o) \approx \sum_{j=1}^k f(\mathbf{p}, \vec{\omega}_o, \vec{\omega}_{i,j}) \frac{\Delta \Phi_j}{A_j} K(|\mathbf{M}_j(\mathbf{p} - \mathbf{x}_j)|), \quad (2.8)$$

where  $k$  represents the number of photons with footprints overlapping the point  $\mathbf{p}$ ,  $\mathbf{x}_j$  is the position of the  $j$ -th photon and  $\Delta \Phi_j$  is its flux,  $A_j$  is the footprint area of the photon,  $K(y)$  is a normalized and symmetric kernel function and  $\mathbf{M}_j$  is a  $3 \times 3$  matrix that performs a change of basis from world space to filter space, where the elliptical footprint becomes a circle with radius one, like in Figure 2.6.

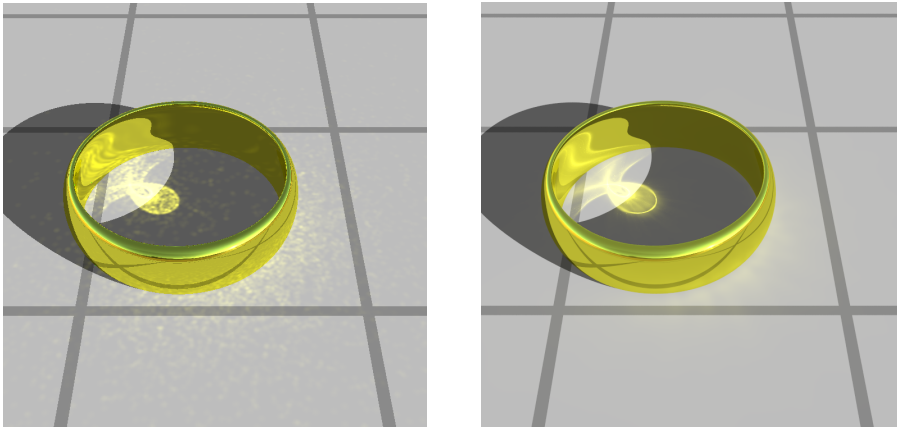
The basis vectors are the positional differential vectors and the surface normal  $\mathbf{n}_j$ :

$$\mathbf{M}_j = \left[ \frac{1}{2} \frac{\partial \mathbf{x}_j}{\partial u} \quad \frac{1}{2} \frac{\partial \mathbf{x}_j}{\partial v} \quad \mathbf{n}_j \right]^{-1}. \quad (2.9)$$



**Figure 2.6:** Change of basis by using the matrix  $\mathbf{M}$ .

By using photon differentials, the bandwidth of the radiance estimate is not constant, but it is defined for each photon and it corresponds to the size of the photon footprint.



(a) Classical photon mapping.

(b) Photon differentials.

**Figure 2.7:** Classical Photon Mapping vs Photon Mapping with photon differentials.

In Figure 2.7 the same scene is rendered once with classical photon mapping and once with photon differentials, in both cases  $5 \cdot 10^5$  photons were used. The caustic in Figure 2.7a is affected by noise while the one obtained by using photon differentials, Figure 2.7b, is almost noise-free and the edges of the caustic are sharp and not blurred as in Figure 2.3b. Photon differentials reduced both noise and bias without increasing the number of photons stored in the photon

maps.



# Volume Rendering

---

The previous chapter provided a description of a global illumination algorithm based on Photon Mapping on the assumption that the light travels through a vacuum, i.e. the radiance changes only when there is an interaction with a surface. This assumption might hold when the light travels through the air; whereas, if another medium is taken into account there are physical processes that affect the propagation of the light and the radiance can not be considered constant anymore. These kinds of media are referred as *participating media*.

The goal of this chapter is to provide a description of how the radiance is affected by participating media and how they can be rendered. First, an introduction of the physical processes that affect the light propagation inside a medium is given. Then, it will be described how Photon Mapping could be extended in order to include participating media and it will be presented an overview of some of the rendering techniques for participating media that have been developed over the years.

## 3.1 Light Transport in a Participating Medium

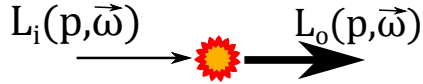
When a beam of light enters a medium, there might be interactions between the photons of the light and the particles of the medium. These interactions can be

of three types: *emission*, *absorption* or *scattering*.

If the properties of these phenomena are constant throughout the volume of the medium, then the medium is called *homogeneous*, otherwise, if the properties change from point to point, the medium is called *heterogeneous*.

### 3.1.1 Emission

The process by which a particle moves from a higher energy state to a lower energy state by emitting a photon and visible light, is called *emission* [ConTCa]. The energy released could come from a chemical, thermal or nuclear process. An example of light emission is provided in Figure 3.2. When a ray travels



**Figure 3.1:** The incoming radiance  $L_i(\mathbf{p}, \vec{\omega})$  is increased by an emission process.

through a medium, the emission process increases the radiance along the ray. This change in radiance is given by the differential equation

$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = L_e(\mathbf{p}, \vec{\omega})dt, \quad (3.1)$$

where  $dt$  represents the differential ray length and  $L_e(\mathbf{p}, \vec{\omega})$  is the emitted radiance per unit of distance at a point  $\mathbf{p}$  in direction  $\vec{\omega}$ .

Equation (3.1) means that the emitted radiance increases linearly with the distance within the medium, so if the ray travels for a distance  $t$  through the medium, the contribution coming from emitted light is  $t \cdot L_e(\mathbf{p}, \vec{\omega})$ .

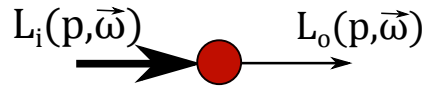
### 3.1.2 Absorption

*Absorption* is the phenomenon by which the energy of a photon travelling through a material is taken up by an atom of the material and converted into internal energy, e.g. thermal energy, [ConTCb].

This process is described by the *absorption coefficient*  $\sigma_a$ , that represents the fraction by which the incoming radiance is reduced per unit of length. This quantity may change with both position and direction, but is usually just a function



**Figure 3.2:** Red and green auroras, Norway [Ols11]. Auroras are the result of the emission of photons in the upper atmosphere.



**Figure 3.3:** The incoming radiance  $L_i(\mathbf{p}, \vec{\omega})$  is reduced by an absorption process.

of the position; the units of measurement are reciprocal distance ( $m^{-1}$ ). If the medium is homogeneous, the absorption coefficient is constant. The differential equation describing the change in radiance along the differential ray length  $dt$  is

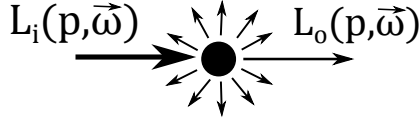
$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = -\sigma_a(\mathbf{p}, \vec{\omega})L_i(\mathbf{p}, \vec{\omega})dt. \quad (3.2)$$

### 3.1.3 Scattering

When a beam of light passes through a medium along a direction  $\vec{\omega}$ , it may collide with particles and be deflected along another direction. This process is called *scattering* and has two effects on the radiance carried by a beam: *out-scattering* and *in-scattering*.

### 3.1.3.1 Out-Scattering

Since some of the photons are scattered away from the ray direction  $\vec{\omega}$ , the exitant radiance  $L_o(\mathbf{p}, \vec{\omega})$  is reduced.



**Figure 3.4:** A fraction of the incoming radiance is scattered along other directions.

As in the absorption case, a parameter called *scattering coefficient*,  $\sigma_s$ , represents the fraction of radiance that is reduced per unit of length due to out-scattering. This phenomenon is described by

$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = -\sigma_s(\mathbf{p}, \vec{\omega})L_i(\mathbf{p}, \vec{\omega})dt. \quad (3.3)$$

Absorption and out-scattering can be combined in one effect called *extinction* or *attenuation*, that represents the total reduction in radiance. The *extinction coefficient*,  $\sigma_t$ , is given by the sum of  $\sigma_a$  and  $\sigma_s$

$$\sigma_t(\mathbf{p}, \vec{\omega}) = \sigma_a(\mathbf{p}, \vec{\omega}) + \sigma_s(\mathbf{p}, \vec{\omega}), \quad (3.4)$$

and the overall radiance attenuation is

$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = -\sigma_t(\mathbf{p}, \vec{\omega})L_i(\mathbf{p}, \vec{\omega})dt. \quad (3.5)$$

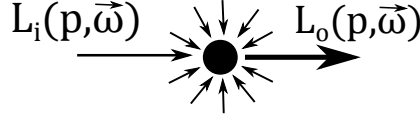
The attenuation, the scattering and the absorption coefficients are usually isotropic and they can be written as  $\sigma_t(\mathbf{p})$ ,  $\sigma_s(\mathbf{p})$  and  $\sigma_a(\mathbf{p})$ .

### 3.1.3.2 In-Scattering

A fraction of the radiance carried by a ray is lost due to out-scattering, on the other hand, photons scattered from other directions may reach the ray and provide a contribution to the radiance. In order to find the in-scattering contribution, the scattered radiance coming from all the possible directions needs to be taken into account:

$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = \sigma_s(\mathbf{p}, \vec{\omega}) \int_{4\pi} p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) L_i(\mathbf{p}, \vec{\omega}') d\vec{\omega}' dt, \quad (3.6)$$





**Figure 3.5:** Radiance is scattered in the direction of the ray.

where  $p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega})$  is called *phase function* and defines the probability of scattering in a certain direction. If the radiance scatters equally in all directions, the phase function is said *isotropic* and it assumes the constant value  $p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) = \frac{1}{4\pi}$ .

The contributions coming from emission and in-scattering can be combined into a function called *source term*,  $S$ :

$$S(\mathbf{p}, \vec{\omega}) = L_e(\mathbf{p}, \vec{\omega}) + \sigma_s(\mathbf{p}, \vec{\omega}) \int_{4\pi} p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) L_i(\mathbf{p}, \vec{\omega}') d\vec{\omega}', \quad (3.7)$$

and then, the total added radiance along the differential length  $dt$  is

$$dL(\mathbf{p}, \vec{\omega}) = L_o(\mathbf{p}, \vec{\omega}) - L_i(\mathbf{p}, \vec{\omega}) = S(\mathbf{p}, \vec{\omega}) dt. \quad (3.8)$$

### 3.1.4 Radiative Transfer Equation

The radiative transfer equation (RTE) was presented by Chandrasekhar [Cha60] and describes the behaviour of light in a medium by accounting for emission, scattering and absorption.

An integro-differential form of the equation can be derived by combining equations (3.5) and (3.8):

$$\frac{\partial L(\mathbf{p}, \vec{\omega})}{\partial t} = -\sigma_t(\mathbf{p}, \vec{\omega}) L_i(\mathbf{p}, \vec{\omega}) + S(\mathbf{p}, \vec{\omega}). \quad (3.9)$$

By integrating (3.9) along a path of length  $s$  from  $\mathbf{p}_0$  to  $\mathbf{p} = \mathbf{p}_0 + s\vec{\omega}$ , the integral form of RTE becomes:

$$\begin{aligned} L(\mathbf{p}, \vec{\omega}) &= T_r(\mathbf{p}_0, \mathbf{p}) L(\mathbf{p}_0, \vec{\omega}) \\ &+ \int_0^s T_r(\mathbf{p}_t, \mathbf{p}) L_e(\mathbf{p}_t, \vec{\omega}) dt \\ &+ \int_0^s T_r(\mathbf{p}_t, \mathbf{p}) \sigma_s(\mathbf{p}_t, \vec{\omega}) \int_{4\pi} p(\mathbf{p}_t, \vec{\omega}' \rightarrow \vec{\omega}) L_i(\mathbf{p}_t, \vec{\omega}') d\vec{\omega}' dt, \end{aligned} \quad (3.10)$$

where  $\mathbf{p}_t = \mathbf{p}_0 + t\vec{\omega}$  are points along the path and  $T_r(\mathbf{p}, \mathbf{p}')$  is the *transmittance* along the segment from  $\mathbf{p}$  to  $\mathbf{p}' = \mathbf{p} + k\vec{\omega}$ :

$$T_r(\mathbf{p}, \mathbf{p}') = e^{-\int_0^k \sigma_t(\mathbf{p}+t\vec{\omega}, \vec{\omega}) dt}, \quad (3.11)$$

where  $k$  is the distance between  $\mathbf{p}$  and  $\mathbf{p}'$ . If the medium is homogeneous, then  $\sigma_t(\mathbf{p}, \vec{\omega}) = \sigma_t$ , and the transmittance can be simplified:

$$T_r(\mathbf{p}, \mathbf{p}') = e^{-\sigma_t \|\mathbf{p} - \mathbf{p}'\|}. \quad (3.12)$$

As in the light transport equation (2.1), RTE does not admit a general solution and typically a solution could be found only by numerical integration.

## 3.2 Related Work

Path tracing [PM93] is one approach to solve the radiative transfer equation, but, as shown in Chapter 2, it has several drawbacks, e.g. slow convergence and noisy results. Photon Mapping is extended into Volumetric Photon Mapping by Jensen and Christensen [JC98] to include participating media. Many researchers have worked on how to improve Volumetric Photon Mapping and some of these works will be presented.

### 3.2.1 Volumetric Photon Mapping

Jensen and Christensen [JC98] propose a method to solve the RTE (3.10) based on Photon Mapping.

The idea behind Volumetric Photon Mapping is to introduce a new photon map, called *volume photon map*, and use it to store photons directly in the volume. Since a photon map is based on a three-dimensional kd-tree, the photons can be stored without changing the underlying algorithms and data structures.

In the *photon pass* the photons are traced from the light sources and if they do not hit a participating medium, the algorithm remains the one presented in Chapter 2. Otherwise, if a photon hits a medium it can either pass through the medium unaffected or can be scattered or absorbed. The probability that a photon interacts with the medium is described by the cumulative probability density function  $F(\mathbf{x})$ :

$$F(\mathbf{x}) = 1 - T_r(\mathbf{x}_0, \mathbf{x}) = 1 - e^{-\int_0^{|\mathbf{x}_0 - \mathbf{x}|} \sigma_t(\mathbf{x}_0 + t\vec{\omega}, \vec{\omega}) dt}, \quad (3.13)$$

where  $\mathbf{x}_0$  is the point where the photon enters the medium.

If the photon interacts with the medium, it is stored in the volume photon map and then Russian Roulette is used to decide whether the interaction is an absorption event, with probability  $\frac{\sigma_a}{\sigma_t}$ , or a scattering event, with probability  $\frac{\sigma_s}{\sigma_t}$ . The quantity  $\frac{\sigma_s}{\sigma_t}$  is called the *scattering albedo*.

When a photon is scattered, a new direction is sampled according to the phase function.

The in-scattered radiance at a point  $\mathbf{p}$ ,  $L_s(\mathbf{p}, \vec{\omega})$ , can be estimated by using the information stored in the volume photon map, but a new definition of radiance has to be found. The estimate presented in Chapter 2 was based on the definition of radiance for surfaces:

$$L = \frac{d^2\Phi}{\cos\theta dA d\omega}, \quad (3.14)$$

where  $\theta$  is the angle between the surface normal and the direction  $\omega$ . When a photon is inside a volume there is no surface normal, and the definition (3.14) cannot be used. Instead of using  $\cos\theta dA$ , the total scattering cross section  $\sigma_s dV$  can be used and the definition of radiance for volumes becomes

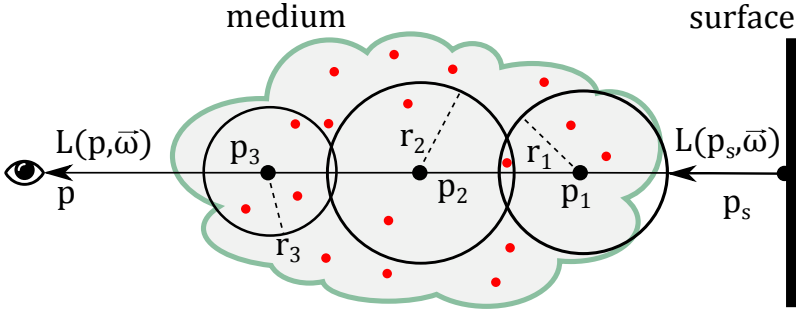
$$L = \frac{d^2\Phi}{\sigma_s dV d\vec{\omega}}, \quad (3.15)$$

and an estimate of the in-scattered radiance at a point  $\mathbf{p}$  is given by

$$\begin{aligned} L_s(\mathbf{p}, \vec{\omega}) &= \int_{4\pi} p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) L_i(\mathbf{p}, \vec{\omega}') d\vec{\omega}' \\ &= \int_{4\pi} p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) \frac{d^2\Phi(\mathbf{p}, \vec{\omega}')}{\sigma_s(\mathbf{p}) dV d\vec{\omega}'} d\vec{\omega}' \\ &\approx \frac{1}{\sigma_s(\mathbf{p})} \sum_{j=1}^N p(\mathbf{p}, \vec{\omega}_j' \rightarrow \vec{\omega}) \frac{\Delta\Phi(\mathbf{p}, \vec{\omega}_j')}{\frac{4}{3}\pi r^3}, \end{aligned} \quad (3.16)$$

where  $r$  represents the radius of the smallest sphere centred at  $\mathbf{p}$  and containing the  $N$  closest photons.

The estimate (3.16) is used in the second pass of Volumetric Photon Mapping. When a ray enters a medium, a *ray marching* algorithm is used to integrate numerically the RTE. Some positions along the ray are sampled and at each point the in-scattered radiance is estimated as in (3.16). By assuming that the emitted radiance in the medium is zero,  $L_e(\mathbf{p}, \vec{\omega}) = 0$ , and that  $\mathbf{p}_s$  is the point on the closest surface reached by the ray, the radiance arriving at a point  $\mathbf{p}$



**Figure 3.6:** The RTE estimate using ray marching.

along the ray can be computed as

$$\begin{aligned}
 L(\mathbf{p}, \vec{\omega}) &= T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \int_0^d T_r(\mathbf{p}, \mathbf{p}_t)\sigma_s(\mathbf{p}_t)L_s(\mathbf{p}_t, \vec{\omega})dt \\
 &\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \sum_{i=1}^S T_r(\mathbf{p}, \mathbf{p}_i) \left( \sum_{j=1}^N p(\mathbf{p}_i, \vec{\omega}_j' \rightarrow \vec{\omega}) \frac{\Delta\Phi_i(\mathbf{p}_i, \vec{\omega}_j')}{\frac{4}{3}\pi r_i^3} \right) \Delta_i,
 \end{aligned} \tag{3.17}$$

where  $d = \|\mathbf{p} - \mathbf{p}_s\|$ ,  $S$  is the number of sampled points along the ray, the step-size  $\Delta_i$  is the distance from  $\mathbf{p}_{i-1}$  to  $\mathbf{p}_i$ , and  $r_i$  is the radius of the smallest sphere centred at  $\mathbf{p}_i$  containing the  $N$  closest photons.

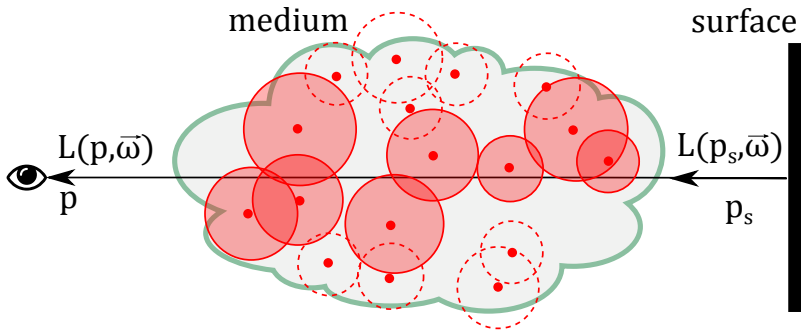
In Figure 3.6 an example is shown: the red dots represent the photons stored inside the photon map, the number of points sampled along the ray is  $S = 3$  and the number of photons used in the in-scattered radiance estimate is  $N = 3$ . The radiance reaching the eye is the sum of the radiance leaving the surface at  $\mathbf{p}_s$  and the in-scattered radiance at  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  and  $\mathbf{p}_3$ .

### 3.2.2 The Beam Radiance Estimate

Volumetric Photon Mapping suffers from the same trade-off between noise and bias presented in Chapter 2. In addition, the ray-marching technique introduces another trade-off between accuracy and computation time: if the step-size is too large, photons might be omitted and, consequently, the result may be noisy; on the other hand, if the step-size is too small, the computation time increases and

it might happen that the same photon is counted more than once because two spheres overlap, like in Figure (3.6).

In both cases the estimate is not accurate, so Jarosz et al. [JZJ08] propose a new estimate, called *the beam radiance estimate*, that eliminates the need for marching through the medium to find photons. The idea is to gather all the photons along the ray in a single query and to use a two-dimensional kernel to blur the radiance estimate (while in the estimate (3.17) the blur is spherical and the kernel is three-dimensional).



**Figure 3.7:** The RTE estimate using the *beam radiance estimate*.

In order to achieve this goal, a kernel bandwidth is associated with each photon and all the photons overlapping the ray are used in the estimate. Jarosz et al. [JZJ08] discuss two approaches to find a kernel bandwidth for each photon: the first approach is to use a constant bandwidth, the second approach is to use a pilot estimate to adapt the kernel width to the local density of photons.

In Figure 3.7 the photons and their bandwidth are represented: the red dots represent the photon positions, the dotted circles are the bandwidths of the photons that are not contributing to the estimate, while the full colored circles are the bandwidths of the photons overlapping the ray.

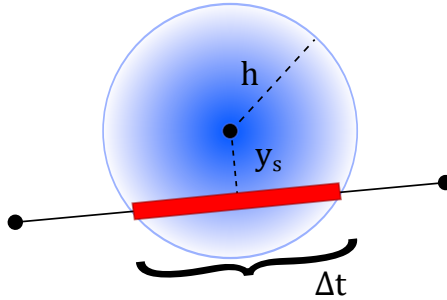
$\mathbf{x}_i$  denoting the position of the  $i$ -th photon,  $h_i$  its bandwidth, and  $\mathbf{p}_i$  the point on the ray that is closest to  $\mathbf{x}_i$ , the RTE can be written as

$$L(\mathbf{p}, \vec{\omega}) \approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) + \sum_{i=1}^N \frac{1}{h_i^2} K\left(\frac{\|\mathbf{p}_i - \mathbf{x}_i\|}{h_i}\right) T_r(\mathbf{p}, \mathbf{p}_i) p(\mathbf{p}_i, \vec{\omega}'_i \rightarrow \vec{\omega}) \Delta\Phi_i(\mathbf{p}_i, \vec{\omega}'_i), \quad (3.18)$$

where  $N$  is the number of photons whose bandwidth overlaps the ray and  $K(y)$  is the kernel function normalized for two dimensions.

### 3.2.3 Photon Differentials

Schjøth [Sch09] uses the same approach as Jarosz et al. [JZJ08] to query the photons along the ray, but, instead of using an expensive pilot estimate to find the bandwidth of each photon, he uses photon differentials to define an anisotropic three-dimensional kernel for each photon. Similarly to Chapter 2, where the footprint of a photon differential was an ellipse, in the case of photon differential for participating media the footprint is in the shape of a three-dimensional ellipsoid. How to compute photon differentials inside a participating medium is described in Chapter 4.



**Figure 3.8:** The contribution given by a photon to a ray depends on the length of the overlapped segment,  $\Delta t$ .

Furthermore, Schjøth proposes to compute the exact contribution of a photon based on the length of the overlap of the photon footprint with the ray. The estimate of the RTE that he proposes is given by

$$L(\mathbf{p}, \vec{\omega}) \approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) + \sum_{i=1}^N \frac{1}{h_i^3} T_r(\mathbf{p}, \mathbf{p}_i) p(\mathbf{p}_i, \vec{\omega}' \rightarrow \vec{\omega}) \Delta \Phi_i(\mathbf{p}_i, \vec{\omega}') w_i, \quad (3.19)$$

where  $N$  is the number of photons overlapping the ray,  $h_i$  is the bandwidth of the  $i$ -th photon,  $\mathbf{p}_i$  is the point on the ray with the shortest distance to the  $i$ -th photon and  $w_i$  is the integrated kernel weight over the length  $\Delta t$  of the overlap:

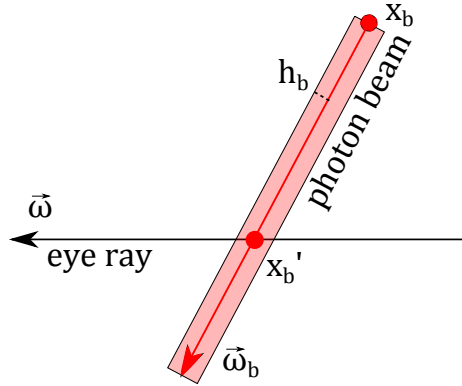
$$w_i = \int_{\Delta t} K\left(\frac{y(t)}{h_i}\right) dt, \quad (3.20)$$

where  $K(y)$  is a normalized and symmetric three-dimensional kernel,  $y(t) = \sqrt{t^2 + y_s^2}$  is the distance from the kernel centre to the line segment that describes

the overlap, and  $y_s$  is the shortest distance between the centre of the kernel and the line segment.

### 3.2.4 Photon Beams

Jarosz et al. [JNSJ11] introduce a new rendering technique for participating media based on *photon beams*. While the rendering techniques presented until now store a photon and its information only when a scattering event happens, the photon beams technique stores the informations regarding the full path covered by a photon.



**Figure 3.9:** Photon beam intersecting an eye ray.

The radiance estimate that they propose is

$$L(\mathbf{p}, \vec{\omega}) \approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) + \sum_{b=1}^B \frac{\sigma_s(\mathbf{p}_b)}{h_b} \frac{p(\mathbf{p}_b, \vec{\omega}_b \rightarrow \vec{\omega})}{\sin(\vec{\omega}_b \cdot \vec{\omega})} \Delta\Phi_b(\mathbf{p}_b, \vec{\omega}_b) T_r(\mathbf{p}, \mathbf{p}_b) T_r(\mathbf{x}_b, \mathbf{x}_b') K\left(\frac{\|\mathbf{x}_b - \mathbf{p}_b\|}{h_b}\right), \quad (3.21)$$

where  $B$  is the number of photon beams intersecting the eye ray,  $h_b$  is the width of the  $b$ -th photon beam,  $\mathbf{p}_b$  is the point on the eye ray that is closest to the  $b$ -th photon beam,  $\mathbf{x}_b$  is the origin of the photon beam and  $\mathbf{x}_b'$  is the point on the photon beam that is closest to the eye ray,  $K(y)$  is a one-dimensional normalized kernel function.

Jarosz et al. [JNT<sup>+</sup>11] extend this approach by introducing progressive photon

beams. They also propose a GPU implementation of the algorithm based on OptiX GPU ray tracing API [PBD<sup>+</sup>10].



In Chapter 3 some of the rendering techniques for participating media that have been developed over the years were presented. By keeping as starting point the work done by Schjøth [Sch09], in this chapter it is described how a more accurate estimate of RTE can be derived by using photon differentials and anisotropic kernels.

The method is divided in two steps as for classical photon mapping: a photon pass and a ray tracing pass. The algorithm is implemented in a GPU environment by using the OptiX ray tracing Engine [PBD<sup>+</sup>10], and the details will be shown in Chapter 5. In the first section of this chapter, the theory that lies behind the photon pass is described, in particular it is explained how photon differentials propagate inside a participating medium, and it is also shown how to trace photons by combining Monte Carlo integration and Russian Roulette. In the second and last section it is presented a new radiance estimate based on photon differentials that works both for isotropic and anisotropic kernels.

## 4.1 Photon Pass

In this section it is first described how to extend the concept of photon differentials from surfaces to volumes, and then are presented the sampling techniques

that are used in the photon pass and an estimate for the flux,  $\Delta\Phi$  associated with each photon.

#### 4.1.1 Photon Differentials for Volumes

In Chapter 2 it was introduced the concept of photon differentials for surfaces, and it was shown how they define a photon footprint in the shape of an ellipse, that can be used to perform a more accurate estimate of the light transport equation.

The details of how to transfer, reflect and refract the photon differentials were not presented, and they can be found in Igehy [Ige99]. Instead, it is useful to describe shortly how an initial value for the positional and directional differential vectors can be found, as proposed by Frisvad et al. [FSES14].

By denoting with  $\mathbf{x}(u, v)$  the position of a photon on the light source, with  $\vec{\omega}(\theta, \phi)$  its direction, and with  $\vec{n}$  the surface normal at  $\mathbf{x}$ , the starting directions of the differential vectors are computed as

$$\frac{\partial_u \mathbf{x}}{|\partial_u \mathbf{x}|} = \frac{\partial_\theta \vec{\omega}}{|\partial_\theta \vec{\omega}|} = \frac{\vec{n} \times \vec{\omega}}{|\vec{n} \times \vec{\omega}|}, \quad (4.1)$$

$$\frac{\partial_v \mathbf{x}}{|\partial_v \mathbf{x}|} = \frac{\partial_u \mathbf{x}}{|\partial_u \mathbf{x}|} \times \vec{n}, \quad \frac{\partial_\phi \vec{\omega}}{|\partial_\phi \vec{\omega}|} = \frac{\partial_\theta \vec{\omega}}{|\partial_\theta \vec{\omega}|} \times \vec{\omega}. \quad (4.2)$$

The sum of all the photon footprint areas must be equal to  $s^2 A_{ls}$ , where  $s$  is a smoothing parameter, and  $A_{ls}$  is the area of the light source. By analogy, the sum of all the photons solid angles should be equal to  $s^2 \omega_{ls}$ , where  $\omega_{ls}$  is the solid angle covered by the light source. If the number of photons emitted from the light source is  $N_e$ , then the initial lengths of the positional and directional differential vectors are computed as

$$|\partial_u \mathbf{x}| = |\partial_v \mathbf{x}| = 2s \sqrt{\frac{A_{ls}}{\pi N_e}}, \quad (4.3)$$

$$|\partial_\theta \vec{\omega}| = |\partial_\phi \vec{\omega}| = 2s \sqrt{\frac{\omega_{ls}}{\pi N_e}}. \quad (4.4)$$

For a point light the value of  $A_{ls}$  is zero, while for a directional light  $\omega_{ls} = 0$ . The parameter  $s$  is used to scale the size of the photon footprint in order to obtain smoother results at the cost of introducing blurring.

Whenever a photon intersects a surface, the surface normal plays an important role in the computation of the new differential vectors and in the computation

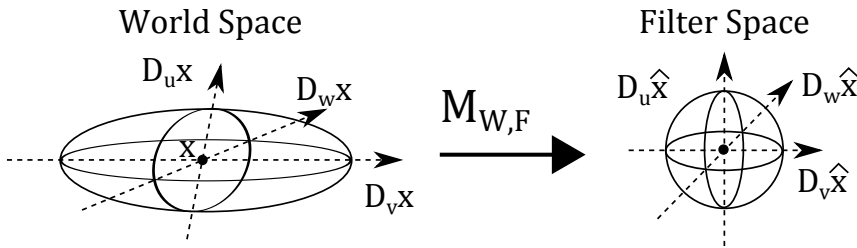
of the matrix  $\mathbf{M}$ , that performs a change of basis from world space to filter space. When participating media are introduced in the scene, the photons are stored inside the medium whenever a scattering event happens, and there is no normal surface that can be used to compute the new differential vectors, so the definition of photon differentials for volumes needs to be expanded.

As proposed by Schjøth [Sch09], a new positional differential vector,  $D_w\mathbf{x}$ , is introduced:

$$D\mathbf{x} = \left\langle \frac{\partial\mathbf{x}}{\partial u}, \frac{\partial\mathbf{x}}{\partial v}, \frac{\partial\mathbf{x}}{\partial w} \right\rangle = \langle D_u\mathbf{x}, D_v\mathbf{x}, D_w\mathbf{x} \rangle, \quad (4.5)$$

where  $D_u\mathbf{x}$  and  $D_v\mathbf{x}$  are the vectors presented in Chapter 2 and they are reflected, refracted and transferred in the same way as before. The third vector is stored and computed only when a photon is inside a participating medium, and it replaces the role of the surface normal.

Schjøth gives to  $D_w\mathbf{x}$  the same direction of the photon at scattering time, and the length is equal to the length of  $D_u\mathbf{x}$  or  $D_v\mathbf{x}$ , whichever is the longest.



**Figure 4.1:** The skewed ellipsoid that represents the kernel volume is transformed into a unit sphere by a change of basis.

The kernel defined by  $D_u\mathbf{x}$ ,  $D_v\mathbf{x}$ , and  $D_w\mathbf{x}$  has the shape of a skewed ellipsoid, whose centre is  $\mathbf{x}$  and the axes are the positional differential vectors. The volume of the ellipsoid is

$$V = \frac{\pi}{6} |(D_u\mathbf{x} \times D_v\mathbf{x}) \cdot D_w\mathbf{x}|, \quad (4.6)$$

while the matrix  $\mathbf{M}_{\mathbf{W},\mathbf{F}}$  that performs the change of basis from world space to filter space, and transforms the skewed ellipsoid into a unit sphere, is

$$\mathbf{M}_{\mathbf{W},\mathbf{F}} = \left[ \frac{1}{2}D_u\mathbf{x} \quad \frac{1}{2}D_v\mathbf{x} \quad \frac{1}{2}D_w\mathbf{x} \right]^{-1}. \quad (4.7)$$

When a photon enters a participating medium, three different events can happen: the photon exits the medium, the photon is scattered or the photon is absorbed.

If the photon is scattered, a new direction  $\vec{\omega}'$  is sampled according to the phase

function, and a new pair of directional differential vectors has to be computed. What Schjøth does is to find the quaternion [ConTCc],  $\mathbf{q}$ , that represents the rotation from the old direction  $\vec{\omega}$  to the new direction  $\vec{\omega}'$ , and to use this quaternion to rotate the differential vectors  $Dx$  and  $D\vec{\omega}$  to the new direction. The quaternion,  $\mathbf{q}$ , is given by

$$\mathbf{q} = \left( \frac{1}{\sqrt{2(1 + \vec{\omega} \cdot \vec{\omega}')}} (\vec{\omega} \times \vec{\omega}'), \frac{\sqrt{2(1 + \vec{\omega} \cdot \vec{\omega}')}}{2} \right). \quad (4.8)$$

In the method proposed in this thesis, only the directional differential vectors  $D\vec{\omega}$  are rotated and the positional differential vectors remain unchanged. The reason behind this choice is that at every interaction with a surface, the differential vectors are transferred onto the surface and then only the directional vectors are modified according to the new direction taken by the photon. It makes sense to apply the same strategy also for volumes, so after transferring a photon differentials to a scattering location, the positional vectors will not be rotated.

## 4.1.2 Sampling Techniques in Participating Media

Every time a photon is traced through a participating medium, there are several events that require a sampling technique. For example, when and where a scattering event is going to happen? Is the photon absorbed or scattered? Which direction will the photon take?

In this section an answer to these questions is provided, and it will also shown how the flux carried by a photon is affected by the sampling techniques.

### 4.1.2.1 Spectrum Sampling

When absorption and scattering were described in Chapter 3, the absorption coefficient  $\sigma_a$  and the scattering coefficient  $\sigma_s$  were presented as function of both the position and direction, but this is not accurate. There is also a third variable that affects these coefficients, and it is the wavelength of the light.

Since the rendered scene follows the RGB color model, a value of  $\sigma_a$  and  $\sigma_s$  is needed for each color channel: red, green, and blue. So, instead of having only

one function to describe absorption and scattering, three functions are needed:

$$\sigma_a(\mathbf{p}, \vec{\omega}) = [\sigma_{a,R}(\mathbf{p}, \vec{\omega}) \quad \sigma_{a,G}(\mathbf{p}, \vec{\omega}) \quad \sigma_{a,B}(\mathbf{p}, \vec{\omega})], \quad (4.9)$$

$$\sigma_s(\mathbf{p}, \vec{\omega}) = [\sigma_{s,R}(\mathbf{p}, \vec{\omega}) \quad \sigma_{s,G}(\mathbf{p}, \vec{\omega}) \quad \sigma_{s,B}(\mathbf{p}, \vec{\omega})], \quad (4.10)$$

$$\sigma_t(\mathbf{p}, \vec{\omega}) = [\sigma_{t,R}(\mathbf{p}, \vec{\omega}) \quad \sigma_{t,G}(\mathbf{p}, \vec{\omega}) \quad \sigma_{t,B}(\mathbf{p}, \vec{\omega})]. \quad (4.11)$$

Having three different functions for each coefficient introduces some difficulties: the cumulative probability density function (3.13), that describes the interaction of a photon with the medium, depends on  $\sigma_t$ , but since there are three different functions describing  $\sigma_t$ , which one of them should be used? And the same problem applies also to the computation of the transmittance  $T_r$ .

The approach that is used in this work to eliminate this problem, is based on Russian Roulette: every time a photon enters a medium one of the three channels (red, green, and blue) is chosen according to a probability density function, and then only the correspondent wavelength is traced through the scene.

For example, if a uniform distribution is used, the extinction coefficient can be written as

$$\sigma_t = \begin{cases} \sigma_{t,R}, & \text{if } \xi < \frac{1}{3} \\ \sigma_{t,G}, & \text{if } \frac{1}{3} \leq \xi < \frac{2}{3}, \\ \sigma_{t,B}, & \text{if } \xi \geq \frac{2}{3} \end{cases}, \quad (4.12)$$

where  $\xi \in ]0, 1]$  is a uniform random variable.

Since also the flux  $\Delta\Phi$  associated with a photon is an RGB value, then it needs to be adjusted. The two wavelengths associated with the discarded color channels are set to zero, while the remaining one is divided by the probability of choosing that wavelength. For example, if for a photon the blue channel is selected, then its flux becomes

$$\Delta\Phi = (\Delta\Phi_R, \Delta\Phi_G, \Delta\Phi_B) \rightarrow \Delta\Phi = \left(0, 0, \frac{\Delta\Phi_B}{pdf(\lambda_B)}\right). \quad (4.13)$$

If the distribution is uniform then  $pdf(\lambda) = \frac{1}{3}$ .

For simplicity, from now on the participating medium is assumed to be homogeneous and isotropic, so that  $\sigma_t(\mathbf{p}, \vec{\omega}) = \sigma_t$ , and analogously for  $\sigma_s$  and  $\sigma_a$ .

#### 4.1.2.2 Scattering Events

Whenever a photon enters a medium it is important to know if it exits the medium or if there is an interaction.

Siegel et al. [SH02] present an estimate of the distance a photon moves before interacting with the medium based on Monte Carlo integration:

$$s = -\frac{\ln(\xi)}{\sigma_t} \quad (4.14)$$

where  $\xi \in ]0, 1]$  is a uniform random variable and  $pdf(s) = \sigma_t T_r(s) = \sigma_t e^{-\sigma_t s}$ .

If the distance  $s$  is lower than the length of the medium, or lower than the distance from the photon origin to the closest surface, then a scattering event has to be sampled. As for the wavelength, also in this case Russian Roulette is used, and the probability for a photon to be scattered is given by the scattering albedo  $\frac{\sigma_s}{\sigma_t}$ , while the probability to be absorbed is  $\frac{\sigma_a}{\sigma_t}$ .

In case of absorption, the photon stops being traced and its flux is set to zero, otherwise in case of scattering, the photon flux is updated and divided by the scattering albedo:

$$\Delta\Phi = \Delta\Phi' \frac{\sigma_t}{\sigma_s}, \quad (4.15)$$

where  $\Delta\Phi'$  is the flux associated with the photon before the scattering event. If the photon is scattered, a new direction is chosen using importance sampling based on the phase function.

#### 4.1.2.3 Flux Estimate

By combining all the sampling techniques presented so far, an estimate of the flux  $\Delta\Phi$  that is associated with a photon, can be done. By denoting with  $\Delta\Phi'$  the flux of a photon before being scattered and with  $\Delta\Phi$  the flux after the scattering event:

$$\Delta\Phi = \int_0^s \sigma_s T_r(t) \int_{4\pi} p(\mathbf{x}_t, \vec{\omega}' \rightarrow \vec{\omega}) \Delta\Phi' d\vec{\omega}' dt, \quad (4.16)$$

by using Monte Carlo integration on the internal integral, the estimate becomes:

$$\Delta\Phi = \int_0^s \sigma_s T_r(t) \frac{1}{M} \sum_{j=1}^M \frac{p(\mathbf{x}_t, \vec{\omega}'_j \rightarrow \vec{\omega})}{pdf(\vec{\omega}'_j)} \Delta\Phi' dt, \quad (4.17)$$

since only one direction,  $\vec{\omega}'_p$  is sampled for each scattering event, then  $M = 1$ . Monte Carlo integration can be used also on the remaining integral:

$$\Delta\Phi = \frac{1}{N} \sum_{i=1}^N \frac{\sigma_s T_r(s_i)}{pdf(s_i)} \frac{p(\mathbf{x}_{s_i}, \vec{\omega}'_p \rightarrow \vec{\omega})}{pdf(\vec{\omega}'_p)} \Delta\Phi'. \quad (4.18)$$

Only one scattering event is sampled according to relation (4.14), and then  $N = 1$ . The estimate (4.18) becomes:

$$\begin{aligned}\Delta\Phi &= \frac{\sigma_s T_r(s)}{\sigma_t T_r(s)} \frac{p(\mathbf{x}_s, \vec{\omega}_p \rightarrow \vec{\omega})}{pdf(\vec{\omega}_p)} \Delta\Phi' \\ &= \frac{\sigma_s}{\sigma_t} \frac{p(\mathbf{x}_s, \vec{\omega}_p \rightarrow \vec{\omega})}{pdf(\vec{\omega}_p)} \Delta\Phi'.\end{aligned}\quad (4.19)$$

Since Russian Roulette is used to select a scattering event (4.15), the flux  $\Delta\Phi'$  has to be divided by the scattering albedo:

$$\begin{aligned}\Delta\Phi &= \frac{\sigma_s}{\sigma_t} \frac{p(\mathbf{x}_s, \vec{\omega}_p \rightarrow \vec{\omega})}{pdf(\vec{\omega}_p)} \Delta\Phi' \frac{\sigma_t}{\sigma_s} \\ &= \frac{p(\mathbf{x}_s, \vec{\omega}_p \rightarrow \vec{\omega})}{pdf(\vec{\omega}_p)} \Delta\Phi'.\end{aligned}\quad (4.20)$$

An additional simplification could be made if the phase function was isotropic, i.e. the photons are scattered equally in all directions. The only possible isotropic phase function is  $p(\mathbf{x}, \vec{\omega}' \rightarrow \vec{\omega}) = \frac{1}{4\pi}$  and  $pdf(\vec{\omega}') = \frac{1}{4\pi}$ . Under this condition, equation (4.20) can be written as:

$$\Delta\Phi = \Delta\Phi'. \quad (4.21)$$

In the end, whenever a photon enters a participating medium and all the sampling techniques described above are used, the only modifications to the flux are done by the spectrum sampling and by importance sampling of the phase function (unless it is an isotropic function).

## 4.2 Ray Tracing Pass

The goal of the Ray Tracing Pass is to trace rays from the eye and to use these rays to estimate the incoming radiance  $L(\mathbf{p}, \vec{\omega})$ , according to the radiative transfer equation (3.10).

The RTE estimate proposed by Schjøth and presented in Chapter 3 is:

$$\begin{aligned}L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s) L(\mathbf{p}_s, \vec{\omega}) \\ &+ \sum_{i=1}^N \frac{1}{h_i^3} T_r(\mathbf{p}, \mathbf{p}_i) p(\mathbf{p}_i, \vec{\omega}'_i \rightarrow \vec{\omega}) \Delta\Phi_i(\mathbf{p}_i, \vec{\omega}'_i) \int_{\Delta t_i} K\left(\frac{y_i(t)}{h_i}\right) dt,\end{aligned}\quad (4.22)$$

where  $\Delta t_i$  is the length of the segment that represents the overlap of the eye ray with the kernel of the photon  $i$ -th, and  $y_i(t)$  is the distance from the  $i$ -th photon to a point on the segment.

Since the kernel function is not the only function that depends on the variable  $t$ , but also the value of the transmittance changes along the segment, this estimate is not completely accurate.

By starting from the RTE, a more precise estimate can be computed:

$$\begin{aligned}
L(\mathbf{p}, \vec{\omega}) &= T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
&+ \int_0^d T_r(\mathbf{p}, \mathbf{p}_t)\sigma_s(\mathbf{p}_t) \int_{4\pi} p(\mathbf{p}, \vec{\omega}' \rightarrow \vec{\omega}) \frac{d^2\Phi(\mathbf{p}_t, \vec{\omega}')}{\sigma_s(\mathbf{p}_t)dVd\vec{\omega}'} d\vec{\omega}' dt \\
&\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
&+ \int_0^d T_r(\mathbf{p}, \mathbf{p}_t)\sigma_s(\mathbf{p}_t) \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{\sigma_s(\mathbf{p}_t)} \frac{1}{h_i^3} K\left(\frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i}\right) dt \\
&\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
&+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \Delta\Phi_i \int_0^d e^{-\sigma_i t} \frac{1}{h_i^3} K\left(\frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i}\right) dt,
\end{aligned} \tag{4.23}$$

where  $N$  is the number of photons which the kernel overlaps the eye ray,  $\mathbf{x}_i$  is the position of the  $i$ -th photon and  $h_i$  its bandwidth,  $d$  is the distance travelled by the eye ray through the participating medium, and  $K(y)$  is a three dimensional normalized and isotropic kernel.

In this thesis the kernel function that is used is the three-dimensional Epanechnikov kernel:

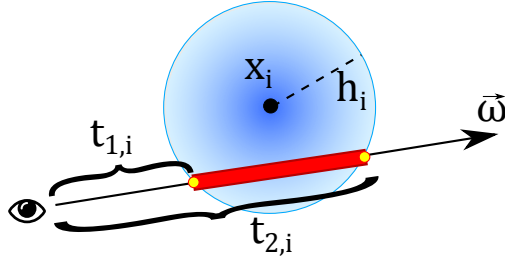
$$K(y) = \begin{cases} \frac{5}{2}(1-y^2)^{\frac{1}{3}} \frac{1}{\pi} & \text{if } |y| < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

Since the kernel function is zero if  $|y| > 1$ , equation (4.23) can be rewritten as

$$\begin{aligned}
L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
&+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_i} \frac{5}{2} \int_{t_{1,i}}^{t_{2,i}} e^{-\sigma_i t} \left(1 - \left(\frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i}\right)^2\right) dt,
\end{aligned} \tag{4.25}$$

where  $V_i = \frac{4}{3}\pi h_i^3$  is the volume of the  $i$ -th photon kernel, and  $t_{1,i}$  and  $t_{2,i}$  are respectively the distances from the ray origin to the points where the ray enters and exits the photon kernel, as in Figure 4.2.





**Figure 4.2:** The eye ray intersects the photon kernel centred at  $\mathbf{x}_i$ , at distances  $t_{1,i}$  and  $t_{2,i}$  from the eye origin. The red segment is the part of the ray that receives a contribution from the photon  $\mathbf{x}_i$ .

The integral in (4.25) can be solved<sup>1</sup> and the estimate becomes

$$\begin{aligned}
 L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s) L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_i} \frac{5}{2} \frac{1}{h_i^2 \sigma_t^2} \\
 &\cdot \left[ (t_{2,i} - t_{1,i}) (e^{-\sigma_t t_{2,i}} + e^{-\sigma_t t_{1,i}}) - \frac{2}{\sigma_t} (e^{-\sigma_t t_{1,i}} - e^{-\sigma_t t_{2,i}}) \right].
 \end{aligned} \tag{4.26}$$

The estimate (4.26) is accurate under the assumption that the kernel is isotropic, i.e. the volume of the kernel is a sphere and the bandwidth  $h_i$  does not change with the direction  $\vec{\omega}$ . Since the shape of a photon differential footprint is a skewed ellipsoid and not a sphere, equation (4.25) needs to be adjusted.

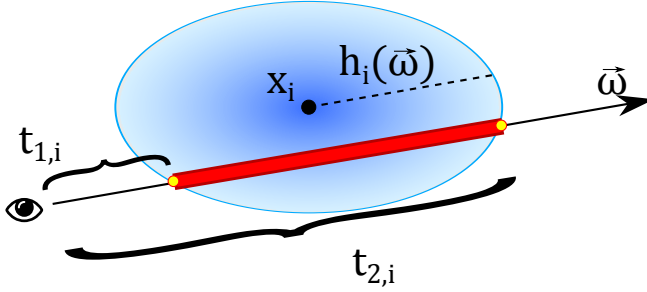
For an anisotropic kernel, the bandwidth is not constant and can be written as a function  $h_i(\vec{\omega}_{i,t})$ , with  $\vec{\omega}_{i,t}$  representing the normalized direction from the kernel centre  $\mathbf{x}_i$  to a point  $\mathbf{p}_t$ :

$$\vec{\omega}_{i,t} = \frac{\mathbf{p}_t - \mathbf{x}_i}{\|\mathbf{p}_t - \mathbf{x}_i\|}. \tag{4.27}$$

Since the matrix  $\mathbf{M}_{\mathbf{W},\mathbf{F}}$  (4.7) describes a linear transformation, and since in filter space the photon differential footprint is a unit sphere, an expression for  $h_i(\vec{\omega}_{i,t})$  can be derived as

$$\|h_i(\vec{\omega}_{i,t}) \mathbf{M}_{\mathbf{W},\mathbf{F}_i} \cdot \vec{\omega}_{i,t}\| = 1 \quad \Rightarrow \quad h_i(\vec{\omega}_{i,t}) = \frac{1}{\|\mathbf{M}_{\mathbf{W},\mathbf{F}_i} \cdot \vec{\omega}_{i,t}\|}. \tag{4.28}$$

<sup>1</sup>The proof is provided in Appendix A.



**Figure 4.3:** The eye ray intersects the anisotropic photon kernel centred at  $\mathbf{x}_i$ , at distances  $t_{1,i}$  and  $t_{2,i}$  from the eye origin. The red segment is the part of the ray that receives a contribution from the photon  $\mathbf{x}_i$ .

By using the anisotropic kernel described by a photon differential footprint, the estimate (4.23) of the RTE becomes:

$$\begin{aligned}
 L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_{ell,i}} \frac{5}{2} \int_{t_{1,i}}^{t_{2,i}} e^{-\sigma_i t} \left( 1 - \left( \frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i(\vec{\omega}_{i,t})} \right)^2 \right) dt \\
 &\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_{ell,i}} \frac{5}{2} \int_{t_{1,i}}^{t_{2,i}} e^{-\sigma_i t} (1 - \|\mathbf{M}_{\mathbf{W}, \mathbf{F}_i}(\mathbf{x}_i - \mathbf{p}_t)\|^2) dt,
 \end{aligned} \tag{4.29}$$

where  $V_{ell,i}$  is the volume of the skewed ellipsoid centred at  $\mathbf{x}_i$ , as described by the equation (4.6).

Similarly to equation (4.26), an exact solution to the integral in (4.29) can be found:

$$\begin{aligned}
 L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) \\
 &+ \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_{ell,i}} \frac{5}{2} \frac{\|\mathbf{M}_{\mathbf{W}, \mathbf{F}_i} \cdot \vec{\omega}\|^2}{\sigma_i^2} \\
 &\cdot \left[ (t_{2,i} - t_{1,i}) (e^{-\sigma_i t_{2,i}} + e^{-\sigma_i t_{1,i}}) - \frac{2}{\sigma_i} (e^{-\sigma_i t_{1,i}} - e^{-\sigma_i t_{2,i}}) \right].
 \end{aligned} \tag{4.30}$$

Equation (4.30) gives a solution to the RTE that is based on photon differentials.

The combination of the photon pass and the ray tracing pass allows to get an accurate representation of a participating medium. Some of the details regarding the implementation on the GPU will be presented in the next chapter.



# Implementation

---

As already said in the previous chapters, the goal of this thesis was to implement an efficient algorithm for rendering participating media based on photon differentials that could run on the GPU.

The *NVIDIA® OptiX™* ray tracing engine [PBD<sup>+</sup>10] is a programmable ray tracing framework, that can be used to increase the speed of ray tracing applications on *NVIDIA® GPUs* using the *NVIDIA® CUDA® GPU* computing architecture. OptiX is a very powerful tool that can be used not only for Computer Graphics rendering, but also for other applications where ray tracing is useful to simulate physical phenomena, e.g. optical and acoustic design, radiation and electromagnetic research, collision analysis.

All the heavy computation in the implementation is handled by the GPU, while the CPU is used to initialize the OptiX programs, to load the scene geometry, and to create several buffers used as input and output by the GPU programs.

In order to reach the final goal, the implementation was divided into three steps:

- **1st step:** the first step was to lay the foundations for progressive photon mapping and photon differentials for surfaces. The photon differential splatting technique presented by Frisvad et al. [FSES14] was used as reference. In order to be able to propagate photon differentials, an interface

supporting both the basic operations between the differential vectors (sum, multiplication by a scalar, dot product, and so on), and the operations described in [Ige99] (transfer, reflection, and refraction) was implemented.

- **2nd step:** the second step was to introduce participating media and volume rendering by using photon differentials. Two different approaches have been tried: in the first attempt, the radiance estimate was based on a ray-marching technique and the data were stored in a kd-tree, but it revealed to be inefficient; the second try was based on the radiance estimate proposed by Schjøth [Sch09] and it was used a more efficient way of storing photons on the GPU.
- **3rd step:** the last goal was to research, and if possible to improve the existing algorithms that make use of photon differentials to render participating media. One improvement of the method proposed by Schjøth [Sch09] has already been made by implementing the algorithm on the GPU, another improvement is the new radiance estimate, equation (4.30), presented in Chapter 4. Unfortunately the time available for the project was not enough to do further research, in Chapter 7 some possible extensions and improvements will be presented.

In the next sections, a description of how the different steps have been implemented is given.

## 5.1 Photon Differential Splatting for Surfaces

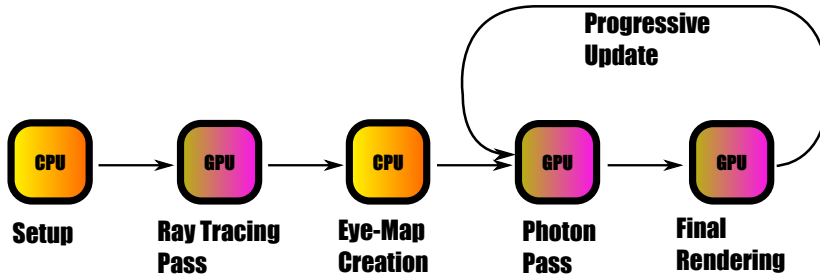
The first step of the project was to implement an algorithm that could render surface caustics by using photon differentials. The technique based on photon differential splatting proposed by Frisvad et al. [FSES14] has been used as reference.

In classical photon mapping, the photons are first stored in a kd-tree, and then, during the ray tracing pass, for each eye path vertex the neighbouring photons are gathered. Instead, in photon differential splatting the ray tracing pass is done first and the eye path vertices are stored in a kd-tree, and then, during the photon pass, each photon distributes its contribution to the eye path vertices covered by the elliptic photon footprint. The final result is the same, but this second approach does not need to store photons and their differentials in a map, and this allows to progressively trace photons without rebuilding a photon map at each pass.

Three different OptiX/CUDA kernels have been implemented in this step: the

first kernel takes care of the ray tracing pass, the second kernel traces photon differentials and splats them, the last kernel combines the results of the previous passes.

The code behind the implementation of photon differential splatting can be divided in five different parts, as illustrated in Figure 5.1:



**Figure 5.1:** CPU and GPU stages of the "Photon Differential Splatting for Surfaces" implementation.

1. **Setup (CPU):** the first part consists only in the initialization of the OptiX programs, of the input and output buffer used to pass data between CPU and GPU, and of the scene.

As already said, three different Ray Generation Programs (CUDA kernels) are defined: the first one is used for the ray tracing pass, the second one for the photon pass, and the third one to compute the final color in each pixel. Three different type of rays are supported: *rtpass rays*, *ppass rays*, and *shadow rays*. A Miss Program and an Exception Program are also defined, and they handles the rays that do not hit any object, and the exceptions.

The size of the ray tracing program and the image resolution are defined by the variables *WIDTH* and *HEIGHT*, while the size of the photon pass is defined by *PHOTON\_WIDTH* and *PHOTON\_HEIGHT*. In order to increase the accuracy of the ray tracing pass, it is also possible to set the number of eye rays traced through each pixel by controlling the variable *pixel\_subdivision* (it is set to 1 by default).

Buffers are used to store results and to exchange data between the CPU and the GPU, the most important of them are: the *Output Buffer* is used to visualize the final rendered result, the *Rtpass-Output Buffer* is used to store the eye vertices and their information after the ray tracing pass, the *Eye-Map Buffer* is used as input for the photon pass and contains the kd-tree with the eye vertices, the *Pixel Buffer* is used to store the contribution coming from direct illumination and from caustic for each pixel, and the

*Image-Random-Seeds Buffer* and the *Photon-Random-Seeds Buffer* contain random numbers that are used to sample points and directions in the two passes of the photon differential splatting algorithm.

The scene includes a light source, and some objects, e.g. *.obj* models and objects defined inside OptiX. The lights that are supported are: *point light*, *area light*, *directional light*, and *spot light*; while the materials and the shaders for the objects are: *diffuse*, *mirror*, and *transparent*.

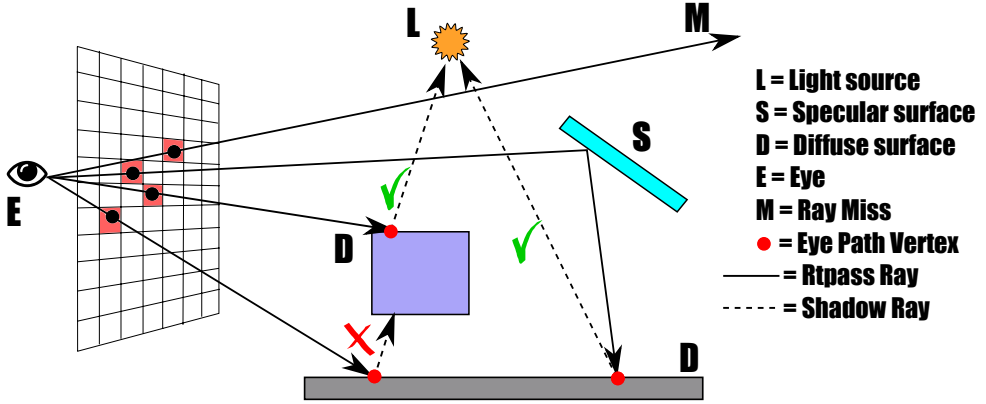
2. **Ray Tracing Pass (GPU):** the first Ray Generation Program that is executed on the GPU is the one corresponding to the Ray Tracing Pass. For each pixel, a number of *rtpass rays* are traced: the origin of each ray is the eye position, and the direction is defined by a normalized vector that goes from the eye to a point inside the pixel.  
 If the ray closest intersection, *hit point*, is on a diffuse surface, then the direct contribution from the light source is computed, and a *shadow ray* is traced towards the light to check for visibility. The position of the hit point, the normal to the surface, the value of the BSDF, and the index of the pixel to which the ray is associated with are stored into the Rtpass-Output Buffer, while the value of the direct illumination is stored into the Pixel Buffer.  
 If the *hit point* is on a specular surface, then a new direction is chosen according to the BSDF associated with the surface material, and a new ray is traced in that direction starting from *hit point*.  
 If a ray does not hit any object, the Miss Program is called and a color, that could be a default value or a value sampled from an environment map, is stored into the Pixel Buffer.
3. **Eye Map Creation (CPU):** after the Ray Tracing Pass, the CPU uses the data stored inside the Rtpass-Output Buffer to create a kd-tree containing the eye path vertices. The information regarding the kd-tree are stored into the Eye-Map Buffer and provided as input to the Photon Pass.
4. **Photon Pass (GPU):** For each photon  $p$ , a position and a direction are sampled according to the type of light source. The photon differentials are computed as described in Section 4.1.1, and each photon carries an amount of flux:

$$\Delta\Phi_p = \frac{\Phi_{ls}}{N_e}, \quad (5.1)$$

where  $\Phi_{ls}$  is the flux of the light source, and  $N_e = PHOTON\_WIDTH \times PHOTON\_HEIGHT$ . Each photon is represented by a *ppass ray*.

If a photon hits a specular surface, the photon differentials are first transferred onto the surface, and then a new direction for the photon is sampled: if the surface is associated with a mirror shader, then the photon direction and the photon differentials are reflected, otherwise, if the surface has a





**Figure 5.2:** Example of Ray Tracing Pass. The red dots represent the points that are stored into the Rtpass-Output Buffer. The shadow rays are traced to check if the light source is visible.

transparent shader, Russian Roulette is used to choose if the photon and its differential vectors should be reflected or refracted.

If a photon hits a diffuse surface after being reflected or refracted, then the photon is first transferred and then splatted. The kd-tree is used to find all the eye path vertices that are inside the elliptical photon footprint: a range-restricted nearest-neighbour search is performed, and the range of the search is the major radius of the photon footprint

$$r_{max} = \frac{1}{2} \max(\|D_u \mathbf{x}_p\|, \|D_v \mathbf{x}_p\|). \quad (5.2)$$

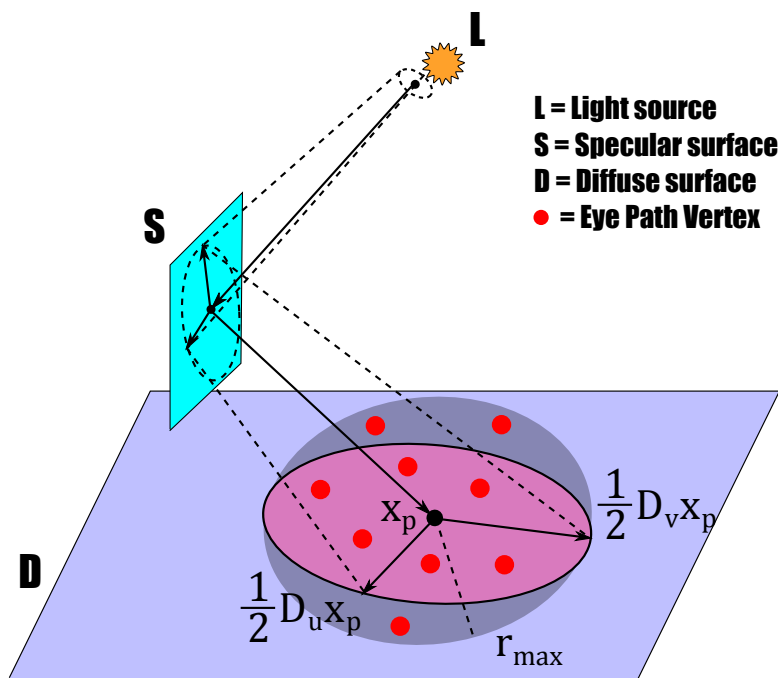
Equation (2.8) can be rewritten to compute the contribution of a single photon:

$$L_r(\mathbf{p}_i, \vec{\omega}_o) \approx f(\mathbf{p}_i, \vec{\omega}_o, \vec{\omega}_{i,p}) \frac{\Delta\Phi_p}{A_p} K(\|\mathbf{M}_p(\mathbf{p}_i - \mathbf{x}_p)\|), \quad (5.3)$$

where  $\mathbf{p}_i$  is the position of the  $i$ -th eye vertex that receives a contribution from the photon  $\mathbf{x}_p$ . The result of (5.3) is evaluated for each eye vertex, and the results are stored into the Pixel Buffer in the position corresponding to the pixel index stored with each eye vertex.

The kernel function  $K(y)$  that is used is the Silvermann's second-order kernel:

$$K(y) = \begin{cases} \frac{3}{\pi}(1-y^2)^2 & \text{for } y < 1 \\ 0 & \text{otherwise} \end{cases}. \quad (5.4)$$



**Figure 5.3:** Example of Photon Pass. A photon and its differential vectors are traced through the scene. When a diffuse surface is hit, all the eye vertices inside the photon footprint are found.

The range-restricted search might return eye path vertices that are not inside the elliptical photon footprint, but since for these vertices  $\|M_p(p_i - x_p)\| > 1$ , the contribution that they receive is null.

Differently from the previous steps that are executed only once, the Photon Pass is executed at every frame. This means that new photons are traced and new splattings are done to improve the accuracy of the result. The final result is computed in the next step.

5. **Final Rendering (GPU):** For each pixel, the Pixel Buffer stores the contribution coming from direct illumination (Ray Tracing Pass) and the contribution coming from the photons (Photon Pass). Since the Photon Pass is executed at every frame, the photon contribution from one frame needs to be combined with the contribution from the previous frames. The sum of direct illumination and photon contribution is then stored into the Output Buffer.

---

**Algorithm 1** Progressive update of the photon contribution.

---

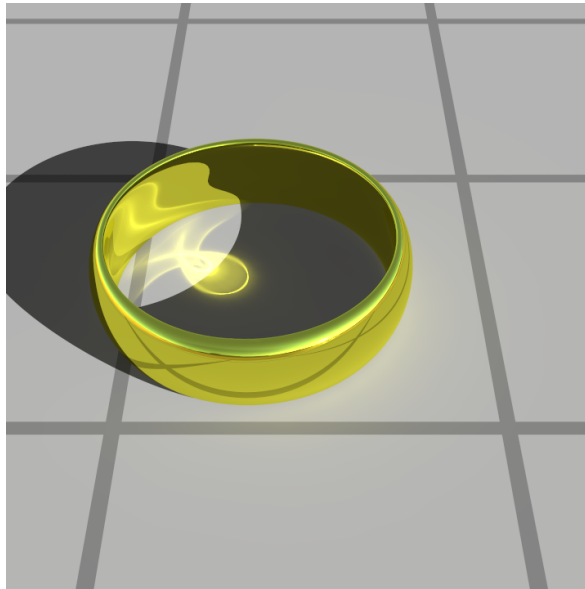
```

a ← 1
b ← 0
if frame_number > 1 then
  a ← 1/frame_number
  b ← (frame_number - 1) * a
end if
photon_contribution ← b * old_contribution + a * new_contribution
old_contribution ← photon_contribution
output_buffer ← direct_illumination + photon_contribution

```

---

Two different results are shown in Figure 5.4 and Figure 5.5. The result in Figure 5.4 shows the cardioid caustic generated by a golden ring illuminated by a directional light. The caustic edges are very sharp, and photon differentials help to remove noise and bias.



**Figure 5.4:** Cardioid caustic on a golden ring.

The result in Figure 5.5 shows a Cornell Box illuminated by a point light. In the scene are present a diffuse block, and a sphere made of glass that projects a caustic on the floor of the Cornell Box.

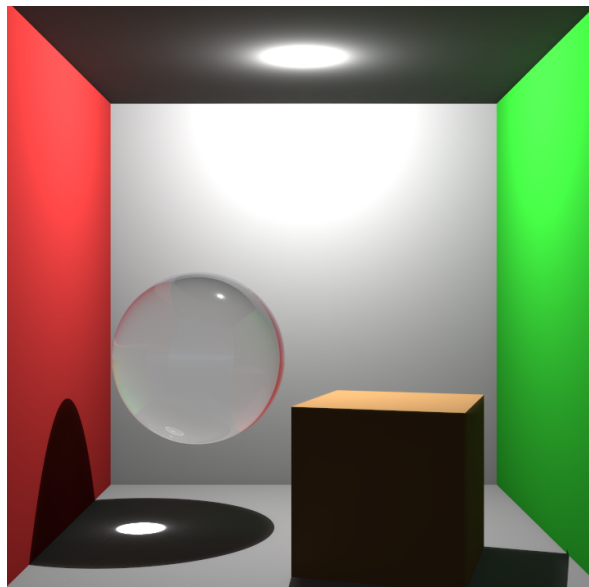


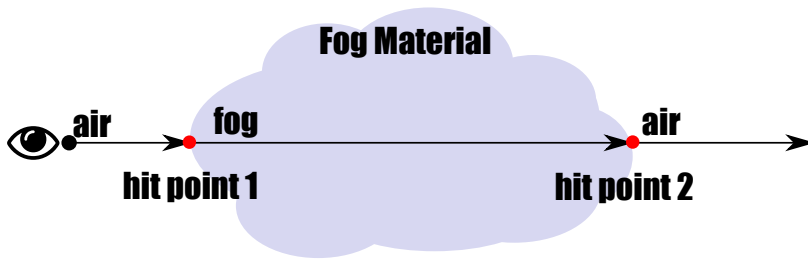
Figure 5.5: Glass sphere and diffuse block inside a Cornell Box.

## 5.2 Volume Rendering of Participating Media Using Photon Differentials

The second task of the project was to extend photon differentials from surfaces to volumes. Two different methods have been tried: the first one is based on the idea of eye-map that has been presented in the previous section; the second one is based on photon map, this means that the Photon Pass precedes the Ray Tracing Pass. The second approach revealed to be more efficient on the GPU.

Regardless of the method chosen, the first thing to do was to define an interface to know whether or not a ray is travelling through a participating medium. In order to do this, a new material, *fog material*, has been implemented; and each ray is associated with an index indicating the medium in which the ray is travelling.

In Figure 5.6 is shown an example: the ray starts outside the medium and the index representative of *air* is associated with it; then the ray hits an object with the *fog material* and, since the ray is entering the object, the index is changed to *fog*; the second time the ray hits the object with the *fog material*, the index is changed to *air*, as the ray is exiting the medium.



**Figure 5.6:** When a ray hits an object with the fog material, the medium index associated with the ray changes depending on whether the ray enters or leaves the object.

### 5.2.1 Volume Rendering based on Eye Map

The first attempt to implement volume rendering was inspired by the method described in Section 5.1 and by Volumetric Photon Mapping: during the Ray Tracing pass, eye vertices are sampled along the rays travelling through a medium by using a ray marching technique, and then they are used to create a kd-tree; during the Photon Pass, every time a photon is scattered, all the eye vertices inside the photon footprint are retrieved from the kd-tree and used to estimate the RTE.

As a ray marching technique is used, this method is very similar to Volumetric Photon Mapping; the main differences are that the bandwidth of the radiance estimate is defined by the photon differential footprint, and the photons are not stored into a photon map but they are traced progressively without building a photon map every frame.

The implementation of this method was thought as a first step towards a more efficient implementation of volume rendering based on the beam radiance estimate, but the passage from points to beams were not trivial with this algorithm. For this reason the method was dropped and a new one, based on a more effective use of the resources offered by OptiX, has been implemented. Nevertheless, it is useful to give a brief description of this first method implementation.

The code follows the same structure of Section 5.1:

1. **Setup (CPU):** the only difference in the Setup step is the addition of two buffers and a variable. The *RtPass-Output Volume Buffer* is used to

store the eye path vertices that are inside a participating medium, and the data contained into it are used to create a kd-tree that will be passed as input to the Photon Pass. The kd-tree is stored into the *Eye-Map Volume Buffer*.

The variable *number\_of\_samples* defines how many points should be sampled and stored into the *RtPass-Output Volume Buffer* for each ray.

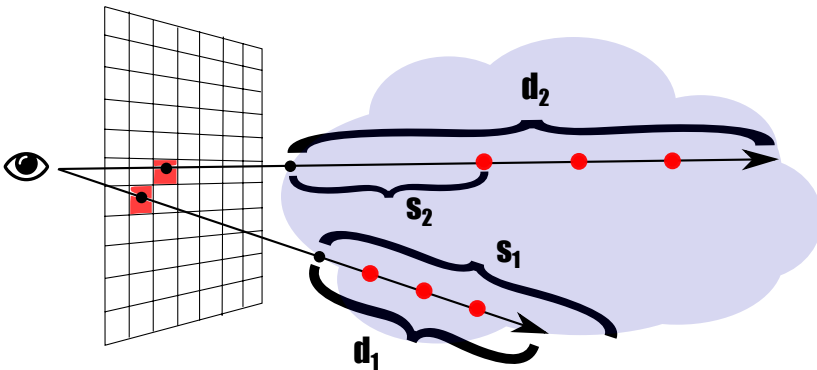
2. **Ray Tracing Pass (GPU):** if a ray does not travel through a medium, the Ray Tracing Pass is the same as described in Section 5.1.

Whenever a ray hits an object with the *fog* material, the medium index of the ray is changed from *air* to *fog* or from *fog* to *air* based on whether the ray is entering or leaving the participating medium. If the ray is entering the medium, a new ray is simply traced from the intersection point between the incoming ray and the object with the *fog* material, and the new direction is the same as the one of the incoming ray.

If the ray is exiting the medium or hits a surface while it is still inside the medium, eye path vertices are sampled along the ray. A random offset, computed as in equation (4.14)

$$s = -\frac{\ln(\xi)}{\sigma_t},$$

is checked against the distance,  $d$ , between the ray origin and the hit point.



**Figure 5.7:** The value of the offset  $s$  is used to define the starting point of the sampling process. The red points represent the eye path vertices that are stored into the eye map.

If the value of  $s$  is lower than the distance  $d$ , then  $d$  is divided into *num-*

## 5.2 Volume Rendering of Participating Media Using Photon Differentials 51

$ber\_of\_samples$  intervals of length

$$\Delta d = \frac{d - s}{number\_of\_samples},$$

and all the points  $\mathbf{p}_i$

$$\mathbf{p}_i = \mathbf{O} + (s + i \cdot \Delta d)\vec{\omega}, \quad \text{for } i = 0, \dots, number\_of\_samples - 1$$

where  $\mathbf{O}$  is the origin of the ray and  $\vec{\omega}$  its direction, are stored into the *RtPass-Output Volume Buffer* along with the index of the pixel associated with the ray.

Otherwise if the value of  $s$  is greater than  $d$ , then the length  $\Delta d$  of the intervals is:

$$\Delta d = \frac{d}{number\_of\_samples + 1},$$

and the points  $\mathbf{p}_i$  that are stored are

$$\mathbf{p}_i = \mathbf{O} + (i \cdot \Delta d)\vec{\omega}, \quad \text{for } i = 1, \dots, number\_of\_samples.$$

An example of ray marching is shown in Figure 5.7.

The reason why the random offset  $s$  is used is to reduce part of the bias, that a fixed number of samples introduces.

After the eye path vertices have been stored, if the ray was exiting the medium, then a new ray is traced starting from the hit point and with a direction equals to the previous direction of the ray; otherwise if the ray hit an object, the material of the object defines the behaviour of the ray, as described in Section 5.1.

3. **Eye Map Creation (CPU):** after the Ray Tracing Pass, the CPU takes care of creating two kd-trees based on the content of the *Rtpass-Output Buffer*, and of the *Rtpass-Output Volume Buffer*. The first kd-tree, stored into the *Eye-Map Buffer*, contains the eye path vertices that are located on surfaces, while the second one, stored into the *Eye-Map Volume Buffer*, contains the eye path vertices that are inside a volume.
4. **Photon Pass (GPU):** The photon and its differential vectors are initialized as in Section 5.1.

If a photon hits a surface without entering a participating medium, then the same steps of section 5.1 are followed.

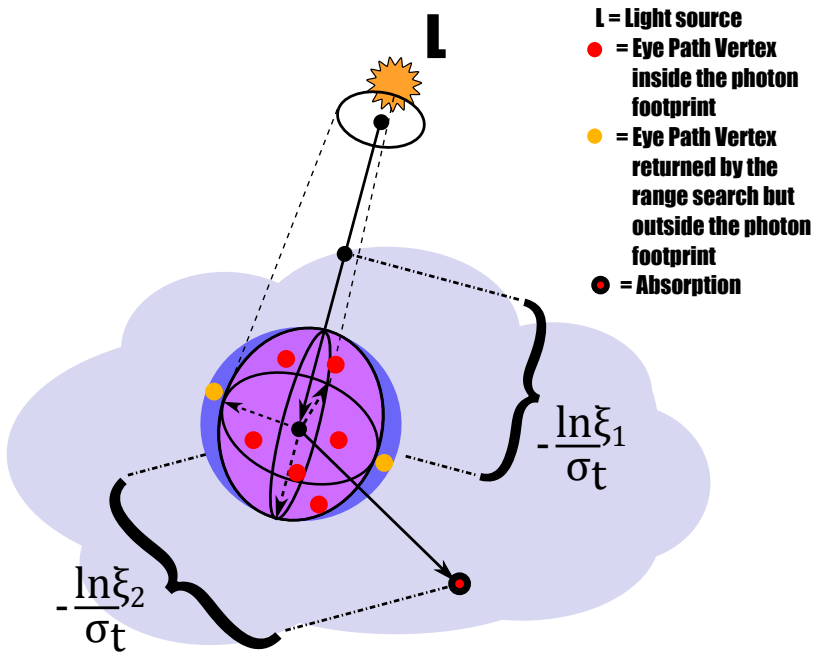
Every time a photon  $p$  enters a medium, one of its color channel is sampled, as described in Section 4.1.2.1, and the index representative of the medium is changed from *air* to *fog*.

When a photon is inside a medium and it hits an object or it is exiting the medium, an offset  $s$  is sampled as in (4.14) and it is compared with

the distance  $d$  between the photon origin and the hit point.

If  $s$  is greater than  $d$ , then the photon hits the object and the hit material defines what happens to the photon as in Section 5.1.

If  $s$  is lower than  $d$ , there is a scattering event and Russian Roulette is used to choose between absorption and scattering: if the result is absorption, the photon is removed from the tracing process; otherwise if the result is scattering, the photon differential vectors are transferred to the scattering position, the photon is splatted, and then traced towards the direction defined by the scattering event.



**Figure 5.8:** A photon is scattered inside a medium, and a range-restricted nearest-neighbour search is performed to find all the eye vertices inside the photon footprint. When the photon is absorbed, it is removed from the Photon Pass.

A range-restricted nearest-neighbour search inside the Eye-Map Volume Buffer is performed in order to look for all the eye path vertices that are inside the photon footprint, as shown in Figure 5.8. The photon footprint is an ellipsoid and it is described by the three positional differential vectors  $D_u\mathbf{x}$ ,  $D_v\mathbf{x}$ , and  $D_w\mathbf{x}$ , as described in Section 4.1.1.

As the length of the differential vector  $D_w\mathbf{x}$  is the longest, the range of



the kd-tree search is:

$$r_{max} = \frac{1}{2} \|D_w \mathbf{x}_p\|. \quad (5.5)$$

The radiance contribution given by a photon  $p$ , at position  $\mathbf{x}_p$ , to the  $i$ -th path vertex, at position  $\mathbf{p}_i$ , is similar to the estimate (3.17) and can be written as

$$L(\mathbf{p}_i, \vec{\omega}) = T_r(\mathbf{p}_O, \mathbf{p}_i) p(\mathbf{p}_i, \vec{\omega}_p' \rightarrow \vec{\omega}) \frac{\Delta \Phi_p}{V_{ell,p}} K\left(\|M_{\mathbf{W}, \mathbf{F}_p}(\mathbf{p}_i - \mathbf{x}_p)\|\right) \Delta d_i, \quad (5.6)$$

where  $\mathbf{p}_O$  is the origin of the ray which the  $i$ -th path vertex belongs to,  $\Delta d_i$  is the length of the interval used to sample the  $i$ -th path vertex,  $V_{ell,p}$  is the volume of the photon footprint, and  $K(y)$  is the Epanechnikov three dimensional kernel function<sup>1</sup>:

$$K(y) = \begin{cases} \frac{5}{2}(1 - y^2) & \text{if } |y| < 1, \\ 0 & \text{otherwise.} \end{cases} \quad (5.7)$$

5. **Final Rendering (GPU):** the final rendering step progressively combines the direct illumination coming from the Ray Tracing Pass and the photon contribution coming from the Photon Pass, as described in Section 5.1.

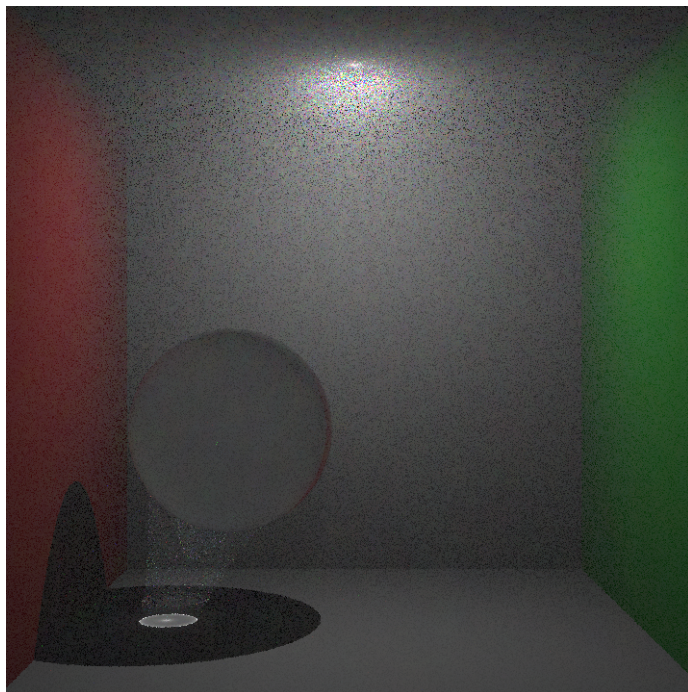
This approach has several downsides, but, as already said, it was thought only as a starting point on which to build a more accurate algorithm.

The biggest issue is due to the use of ray marching: as explained in Chapter 3, the number of points sampled on a ray introduces a trade-off problem between noise and performance. Since the eye vertices are stored into a kd-tree, and the Ray Tracing Pass is executed only once, if only few points are sampled with ray marching then the result will be very noisy, and even if a very high number of photons are traced, the noise will not be completely removed. On the other hand, if too many eye vertices are stored into the volume eye-map, the creation of the kd-tree and the Photon Pass will require more time to be performed, and the radiance estimate will still be inaccurate for the reasons explained in Chapter 3.

In Figure 5.9 it is shown a result obtained with this method: the scene contains a Cornell Box and a glass ball, and both are placed inside a participating medium; ten eye vertices are sampled for each ray by using ray marching, and it is possible to see how with only few vertices per ray most of the photons are missed (in particular inside the volume caustic and close to the point light).

One possible way to remove part of the noise would be to run a Ray Tracing Pass

<sup>1</sup>This version of the Epanechnikov kernel function is not normalized, as the estimate (5.6) is already divided by the volume of the footprint.



**Figure 5.9:** Cornell Box and a glass sphere inside a participating medium, with ten eye path vertices sampled for each ray. Ray marching introduces noise that is very visible inside the volume caustic and close to the point light.

at each frame, in this way the eye path vertices used in the radiance estimate will be different at every pass. Unfortunately the construction of the kd-tree containing the volume eye-map is expensive, and if executed every frame it will considerably reduce the performance.

The best solution would be to remove ray marching and to introduce an estimate based on the full path of the eye ray and not only on some sampled points, e.g. the beam radiance estimate or the method proposed by Schjøth. In order to do this, the eye ray should be stored into the volume eye-map during the Ray Tracing Pass, and whenever a photon is splatted, all the beams passing through the photon footprint should be retrieved from the kd-tree. Unfortunately the computational cost of building a kd-tree at each frame would not be removed, and switching from a kd-tree containing points to one that contains lines would not be straightforward.

As the use of an eye-map turned out to be inefficient for rendering volumes, this method has been discarded and a new one, based on photon maps, has been

implemented.

### 5.2.2 Volume Rendering based on Photon Map

The method just described was based on the same structure of the Photon Differential Splatting for surfaces: during the Ray Tracing Pass all the eye path vertices were found and stored into a buffer, then the kd-tree was built on the CPU, and a progressive Photon Pass was performed. The amount of eye vertices stored into the kd-tree introduced a trade-off between noise and performance that could not be easily removed by using an eye-map.

The new approach described in this section is based on photon map, i.e. the Photon Pass is executed first and a photon map is created, and then during the Ray Tracing Pass all the photons with footprint overlapping the eye ray are found and their contribution is computed by using the estimate (3.19) proposed by Schjøth [Sch09].

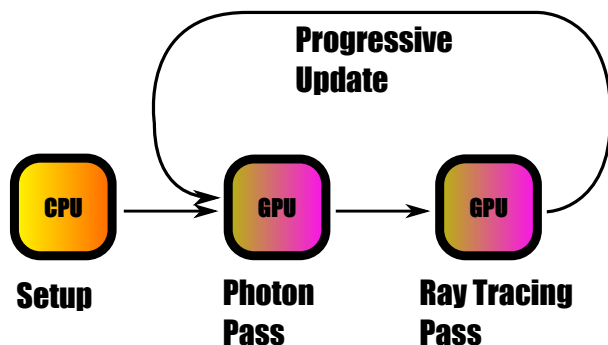
The choice of using a kd-tree as data structure works well if the range search is performed to search points, but if lines representing eye rays need to be found, then a kd-tree is not the best choice. Moreover, the kd-tree has to be built on the CPU after that the Photon Pass has been completed, and if the amount of data to be stored is large, then it might be too expensive to create a photon map at every frame. For these reasons, a new way of storing the photon map, based on a better exploit of OptiX, is used.

The idea is to store the photons as geometry during the Photon Pass, and to gather all the photons with footprint overlapping the eye ray by simply intersecting the ray with the photon geometry.

The implementation of the method can be divided into three main parts, as illustrated in Figure 5.10:

1. **Setup (CPU):** in this approach only two Ray Generation programs are used, one for tracing photons, and one for tracing eye rays. A new type of ray called *splatting ray* is introduced compared to the previous implementations, and it is used to intersect all the photons that are placed along a *rtpass ray*.

The *Output Buffer*, the *Photon-Random-Seeds Buffer*, and the *Image-Random-Seeds Buffer* are still used in this implementation. The *Photon Geometry Buffer* is a new buffer, and it contains information regarding the geometry of each photon, i.e. the position and the maximum radius of the footprint, the shape of the footprint (disk, ellipsoid, or sphere), and a directional component (the normal surface if the photon is on a surface, or the direction of the photon at scattering time if the photon is inside a vol-



**Figure 5.10:** CPU and GPU stages of the "Volume Rendering based on Photon Map" implementation.

ume). The *Photon Data Buffer* is the second new buffer introduced in this implementation, and it is connected with the Photon Geometry Buffer; it contains information useful to compute the radiance contribution of each photon: the carried flux  $\Delta\Phi$ , the differential vectors  $D_u\mathbf{x}$ ,  $D_v\mathbf{x}$ ,  $D_w\mathbf{x}$ , the matrix  $\mathbf{M}_{\mathbf{W},\mathbf{F}}$ , and the size of the photon footprint.

The sizes of the Photon Geometry Buffer and of the Photon Data Buffer are equal to the size of the Photon Pass Ray Generation program,  $PHOTON\_WIDTH \times PHOTON\_HEIGHT$ , in this way each photon is associated to a position within the buffers.

Each element of the Photon Geometry Buffer is associated with a Bounding Box, an Intersection Program, and an Any Hit Program that are called every time the photon is intersected by a *splatting ray*.

The information stored inside the buffers are used to create an OptiX Geometry Group representing the photon map, and an acceleration structure is used to improve the performance: a *TRBVH* (Treelet Reordering Bounding Volume Hierarchy) builder is used to build the photon map, and a *BVH* (Bounding Volume Hierarchy) is used to traverse it<sup>2</sup>. The use of a TRBVH builder allows to have a very fast build of the Geometry Group so that the photon map can be updated and rebuilt after every Photon Pass without any performance issue. While the use of a BVH to traverse the photon map allows to quickly find the photons intersected by a ray.

The Geometry Group storing the photon map is separated from the Geometry Group containing the scene, and it is affected only by *splatting rays*, this means that the other ray types do not intersect the photon geometry.

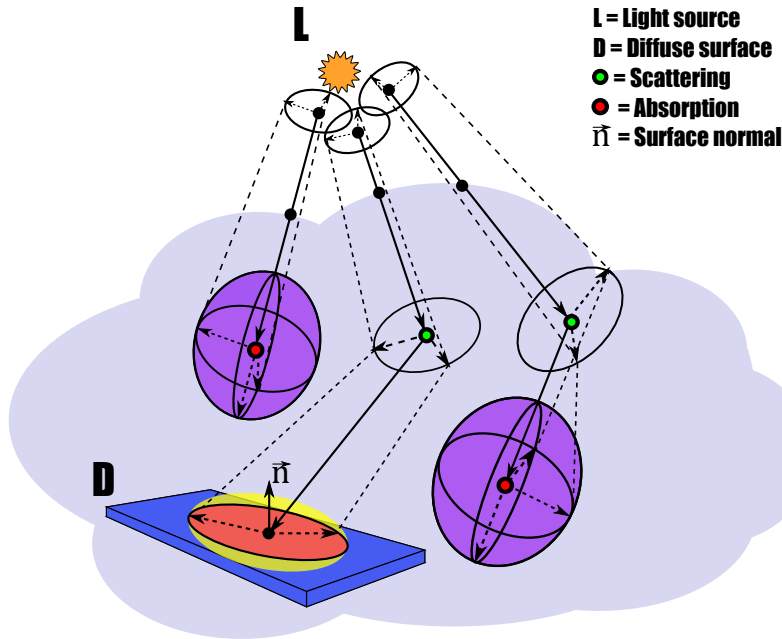
<sup>2</sup>More information regarding the mode of operation of OptiX can be found in the OptiX Documentation [CorTC].

## 5.2 Volume Rendering of Participating Media Using Photon Differentials 57

2. **Photon Pass (GPU):** each time a photon  $p$  is initialized during the Photon Pass, the information stored in the corresponding positions of the Photon Geometry Buffer and Photon Data Buffer are all reset to a default value. The flux, the values of the photon differentials, the position on the light source and the starting direction are computed as in Section 5.1.

If a photon enters a participating medium, Spectrum Sampling is used to choose one of the color channel carried by the photon.

Whenever a photon is inside a medium and intersects an object, an offset  $s$  is computed as in equation (4.14) and compared with the distance  $d$  between the photon origin and the hit point. If  $s$  is lower than  $d$ , then a scattering event has to be sampled by using Russian Roulette, otherwise the path of the photon continues unaltered.



**Figure 5.11:** Example of photon geometry creation using ellipsoids and disks. The photons are stored as ellipsoids whenever an absorption is sampled, or as disk (the yellow shape) whenever a diffuse surface is hit.

In case of scattering, the photon differential vectors are transferred to the scattering position, a new direction is sampled, the directional differential vectors are rotated towards the new direction, as described in Chapter 4,

and the photon is traced towards the new direction.

In case of absorption, the photon differential vectors are transferred to the scattering position, and then the photon is stored into the photon map: the scattering position  $\mathbf{x}_p$ , the length of the positional differential vector  $D_w \mathbf{x}_p$ , the direction of the photon  $\vec{\omega}_p$  before the absorption, and a flag representing the shape of the photon footprint (a sphere if an isotropic kernel is used, or an ellipsoid if an anisotropic kernel is used) are stored into the Photon Geometry Buffer, while the value of the positional differential vectors, the matrix of change of basis from world space to filter space, the flux carried by the photon, and the volume of the photon footprint are stored into the Photon Data Buffer.

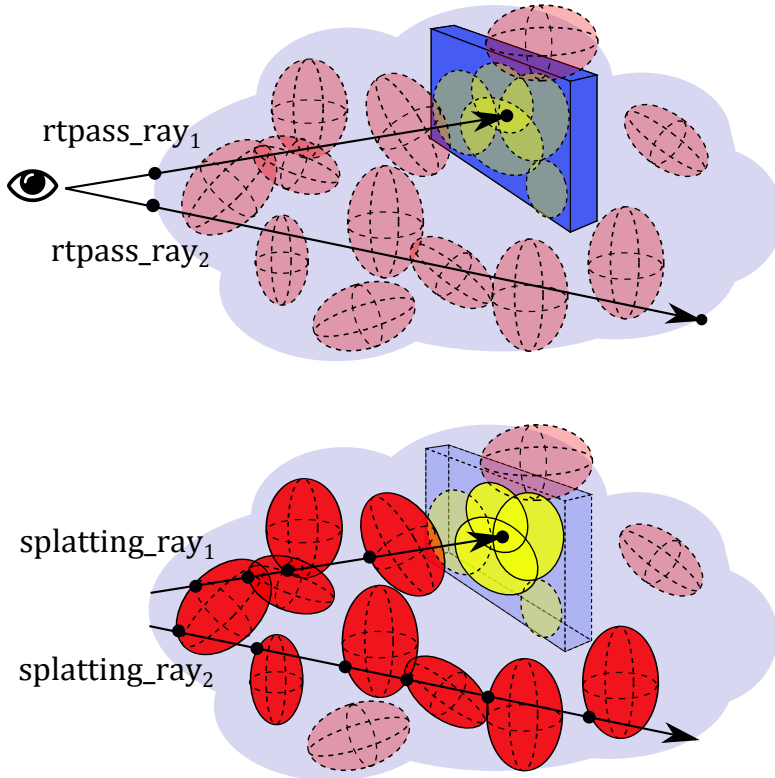
Since only one element of the Photon Geometry Buffer is available for each photon, the fact of using the absorption to define which event should be saved into the photon map helps to reduce the bias that will be introduced by storing always the same scattering event, e.g. always the first one or always the second one.

If the photon hits a surface, then the material of the surface defines the behaviour of the photon. If the material is specular, the photon is reflected or refracted and its photon differential vectors are updated accordingly, otherwise if the material is diffuse, the photon is stored into the photon map: the hit point  $\mathbf{x}_p$ , the length of the longest positional differential vector, the normal  $\vec{n}$  of the surface at position  $\mathbf{x}_p$ , and a flag representing the disk shape are stored into the Photon Geometry Buffer, while the value of the positional differential vectors, the matrix of change of basis from world space to filter space, the flux carried by the photon, and the area of the photon footprint are stored into the Photon Data Buffer. The reason why a disk is used to define the shape of a photon footprint on a surface instead of an ellipse is that it is easier to intersect a ray with a disk than with an ellipse, and anyway all the points that are inside the disk but outside the ellipse will give a null contribution to the radiance estimate 5.3. In Figure 5.11 an example of this process is shown.

When the Photon Pass is completed, the Geometry Group containing the photon map is mark as dirty, in this way the acceleration structure is rebuilt and updated before the next Ray Generation Program is launched.

3. **Ray Tracing Pass (GPU):** the mode of operation of the Ray Tracing Pass is quite simple. Whenever a *rtpass ray* enters a medium, a new *rt-pass ray* is traced in the same direction of the incoming ray, and the index representing the medium is changed from *air* to *fog*.  
If a ray hits a surface or is exiting a medium, then a *splattting ray* is traced from the origin of the *rtpass ray* to the hit point (a  $t_{max}$  value is used to set the length of the *splattting ray*). The *splattting ray* intersects all the photons, with footprint overlapping the *rtpass ray*, that are located inside the volume of the medium or onto the hit surface. This process is shown

in Figure 5.12.



**Figure 5.12:** When a *rtpass ray* hits a surface or it is exiting a medium, a *splatting ray* is traced to find all the photons that are placed along the ray path. The *rtpass rays* do not intersect the photons, while the *splatting rays* do not intersect the scene geometry. The intersection points are shown as black dots.

When a *splatting ray* intersects the Bounding Box of a photon, the Intersection Program of the photon is called and it is associated with a unique index that is used to access the Photon Geometry Buffer, and the Photon Data Buffer.

If the shape of the photon is a disk, then the information stored into the Photon Geometry Buffer are used to define the centre, the radius, and the normal vector of the disk, and a simple ray-disk intersection is performed. If the shape of the photon is a sphere, then the photon position and the

maximum radius of the photon footprint are used to define the centre and the radius of the sphere, and a ray-sphere intersection is computed.

If the shape is an ellipsoid, the matrix  $\mathbf{M}_{\mathbf{W},\mathbf{F}}$  is used to pass from world space to filter space, in this way the ellipsoid becomes a unit sphere and a ray-sphere intersection can be performed; the result of the ray-sphere intersection is then multiplied by the inverse matrix  $\mathbf{M}_{\mathbf{F},\mathbf{W}}$  in order to obtain the intersection points in world space.

If the Intersection Program finds an intersection, then the Any Hit Program associated with the *splatting ray* is called, and the index of the intersected photon, and the intersection points are passed as parameter. The goal of the Any Hit Program is to compute the radiance contribution given by the intersected photon to the eye ray.

If the intersected photon is a disk, then the estimate (5.3) is computed.

If the intersected photon is a sphere or an ellipsoid, then the contribution given by the photon  $p$  to the  $i$ -th *splatting ray*, is computed as proposed by Schjøth [Sch09]:

$$L(\mathbf{p}_i, \vec{\omega}_i) \approx \frac{1}{h_p^3} T_r(\mathbf{p}_i, \mathbf{p}_p) p(\mathbf{p}_p, \vec{\omega}_p' \rightarrow \vec{\omega}_i) \Delta\Phi_p w_p, \quad (5.8)$$

where  $\mathbf{p}_i$  and  $\vec{\omega}_i$  are the origin and the direction of the *splatting ray*,  $\mathbf{p}_p$  is the point on the *splatting ray* with the shortest distance from the photon position  $\mathbf{x}_p$ ,  $w_p$  is the integrated kernel weight described in equation (3.20). The kernel function that is used is the normalized three-dimensional Epanechnikov kernel (4.24), and the value of  $w_p$  depends on the shape of the photon footprint. If the photon footprint is a sphere, the kernel function is isotropic and the estimate (5.8) becomes

$$L(\mathbf{p}_i, \vec{\omega}_i) \approx T_r(\mathbf{p}_i, \mathbf{p}_p) p(\mathbf{p}_p, \vec{\omega}_p' \rightarrow \vec{\omega}_i) \frac{\Delta\Phi_p}{V_p} \frac{10}{3h_p^2} \sqrt{(h_p^2 - \|\mathbf{p}_p - \mathbf{x}_p\|^2)^3}, \quad (5.9)$$

where  $V_p$  is the volume of the photon footprint. If the shape of the footprint is an ellipsoid, the estimate is

$$L(\mathbf{p}_i, \vec{\omega}_i) \approx T_r(\mathbf{p}_i, \mathbf{p}_p) p(\mathbf{p}_p, \vec{\omega}_p' \rightarrow \vec{\omega}_i) \frac{\Delta\Phi_p}{V_p} \cdot \frac{10}{3\|\mathbf{M}_{\mathbf{W},\mathbf{F}} \cdot \vec{\omega}_i\|} \left(1 - \|\mathbf{M}_{\mathbf{W},\mathbf{F}} \cdot (\mathbf{p}_p - \mathbf{x}_p)\|^2\right)^{\frac{3}{2}}. \quad (5.10)$$

As the Any Hit Program is executed for all the photons intersected by the *splatting ray*, the exact contribution of equation (3.19) is computed.

After that the *splatting ray* has been traced, the Ray Tracing Pass resumed from where it was interrupted, and the material of the object that was hit by the *rtpass ray* defines if a new *rtpass ray* should be traced or if direct



illumination should be computed.

The Ray Tracing Pass takes also care of progressively updating the final color of each pixel by combining the direct illumination and the radiance computed with *splatting rays*.

This approach provides a more accurate result than the one described in the previous section. The use of a Geometry Group allows to build a photon map very quickly and to gather photons by simply tracing a ray. Since the Photon Pass and the Ray Tracing Pass are executed at every frame, the result is progressively refined.

The next part of the thesis work was done based on this implementation of volume rendering.

### 5.3 Research and Improvement

The final part of the thesis work has been dedicated to research and improve the existing algorithms that make use of photon differentials for rendering volumes. As already said, one improvement consists in the GPU implementation of the method presented by Schjøth [Sch09], and unfortunately the amount of time that was left for this part of the project has been sufficient only to compute and implement a more accurate estimate of the RTE than the one presented by Schjøth.

The implementation of the new radiance estimate follows the same step of Section 5.2.2, the only difference is that equation (5.9) and equation (5.10) are replaced by the estimates that were presented in Chapter 4.

If the photon footprint is a sphere, the radiance is computed as in equation (4.26), and the contribution given by a single photon  $p$  to the  $i$ -th *scattering ray* becomes

$$L(\mathbf{p}_i, \vec{\omega}_i) \approx p(\mathbf{p}_p, \vec{\omega}'_p \rightarrow \vec{\omega}_i) \frac{\Delta\Phi_p}{V_p} \frac{5}{2} \frac{1}{h_p^2 \sigma_t^2} \cdot \left[ (t_{2,p} - t_{1,p}) (e^{-\sigma_t t_{2,p}} + e^{-\sigma_t t_{1,p}}) - \frac{2}{\sigma_t} (e^{-\sigma_t t_{1,p}} - e^{-\sigma_t t_{2,p}}) \right]. \quad (5.11)$$

where  $\mathbf{p}_i$  and  $\vec{\omega}_i$  are the origin and the direction of the *scattering ray*,  $\mathbf{p}_p$  is the point on the *splatting ray* with the shortest distance from the photon position  $\mathbf{x}_p$ ,  $\Delta\Phi_p$  is the flux carried by the photon  $p$ ,  $V_p$  and  $h_p$  are the volume and the radius of the spherical photon footprint,  $t_{1,p}$  and  $t_{2,p}$  are the distances from the origin of the *scattering ray* to the first and second intersections with the photon

footprint.

If the shape of the photon footprint is an ellipsoid, the radiance estimate is the one presented in equation (4.30), and the photon contribution becomes

$$L(\mathbf{p}_i, \vec{\omega}_i) \approx p(\mathbf{p}_p, \vec{\omega}'_p \rightarrow \vec{\omega}_i) \frac{\Delta\Phi_p}{V_{ell,p}} \frac{5}{2} \frac{\|\mathbf{M}_{\mathbf{W},\mathbf{F}_p} \cdot \vec{\omega}_i\|^2}{\sigma_t^2} \cdot \left[ (t_{2,p} - t_{1,p}) (e^{-\sigma_t t_{2,p}} + e^{-\sigma_t t_{1,p}}) - \frac{2}{\sigma_t} (e^{-\sigma_t t_{1,p}} - e^{-\sigma_t t_{2,p}}) \right]. \quad (5.12)$$

where  $V_{ell,p}$  is the volume of the photon footprint,  $\mathbf{M}_{\mathbf{W},\mathbf{F}_p}$  is the matrix of change of basis from world space to filter space, and  $\frac{1}{\|\mathbf{M}_{\mathbf{W},\mathbf{F}_p} \cdot \vec{\omega}_i\|}$  is the radius of the ellipsoid along the direction  $\vec{\omega}_i$ .

In this chapter the results obtained with the methods implemented during this project are shown.

First some simple results are presented in order to show how absorption and scattering affect the propagation of light inside a medium. Then it will be made a comparison between the use of anisotropic kernel functions and the use of isotropic functions, in order to prove the efficacy of photon differentials for rendering volumes. Finally some test scenes are used to compare the results obtained by using the radiance estimate presented in Chapter 4 and the results obtained with the estimate proposed by Schjøth [Sch09].

All the test scenes include an isotropic and homogeneous participating medium, since there was no time to implement the code for heterogeneous media. The versions of OptiX and CUDA used in this project are the OptiX SDK 3.7.0 and CUDA v6.5, while the processor used to run the code is an Intel® Core™ i7-4700MQ 2.40GHz 16GB, and the GPU is a GeForce GT 740M.

## 6.1 Absorption and Scattering

In Chapter 3 it was discussed how participating media affect the propagation of light. It was explained how absorption reduces the energy of the photons trav-

elling through a medium, and how the scattering of light could both decrease (out-scattering) or increase (in-scattering) the amount of radiance.

Some test scenes have been rendered in order to show these two different phenomena. The test scenes contain a directional light, a neutral background, and a 3D model used as participating medium. The 3D model that have been used are .OBJ files and they are an elephant model, and the Stanford dragon. All the scenes were rendered for ten seconds at a frame rate of 15 fps. The implementation used is the one described in Section 5.3.

If a participating medium primarily absorbs light, the medium will have a dark appearance.



**Figure 6.1:** Participating medium with the shape of the elephant model. The light is mainly absorbed by the medium and the volume looks dark.

In Figure 6.1 and in Figure 6.2 two examples of absorption in participating media are shown. In the first scene the 3D model of the elephant is used to represents the medium, while in the second one it is used the Stanford dragon. It is very visible the exponential relationship between the amount of light absorbed and the thickness of the medium: close to the edges of the Stanford dragon and in the ears and trunk of the elephant there is less absorption, while by moving towards the centre of the medium the absorption increases very quickly.

If the dominant effect in a medium is scattering, then the medium will still absorb light but it will have a brighter appearance.

In Figure 6.3 and in Figure 6.4 two examples of light scattering in participating media are shown. In Figure 6.3 the medium has been rendered with a smaller



**Figure 6.2:** Participating medium with the shape of the Stanford dragon. The light is mainly absorbed by the medium and the volume looks dark.



**Figure 6.3:** Participating medium with the shape of the elephant model. The light is mainly scattered and the medium looks more brighter than the background.

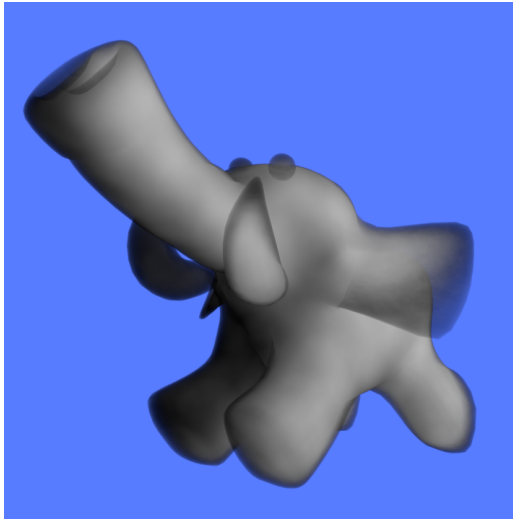
value of the extinction coefficient  $\sigma_t$  than the medium in Figure 6.4. It is visible how light scattering increases the radiance of the light. In the elephant picture



**Figure 6.4:** Participating medium with the shape of the Stanford dragon. The light is mainly scattered and the medium looks more brighter than the background.

part of the light passes through the medium and the background it is visible; on the other hand the extinction coefficient used in the Stanford dragon result is large and the light coming from the background can not pass through the medium, but part of the light coming from the source light is scattered towards the camera and the dragon assumes a white appearance.

In Figure 6.5 and 6.6 two participating media with a balanced amount of scattering and absorption are shown. The results are brighter than the ones in Figure 6.1 and 6.2 but darker than the results in Figure 6.3 and 6.4.



**Figure 6.5:** Participating medium with the shape of the elephant model. Absorption and scattering are balanced.



**Figure 6.6:** Participating medium with the shape of the Stanford dragon. Absorption and scattering are balanced.

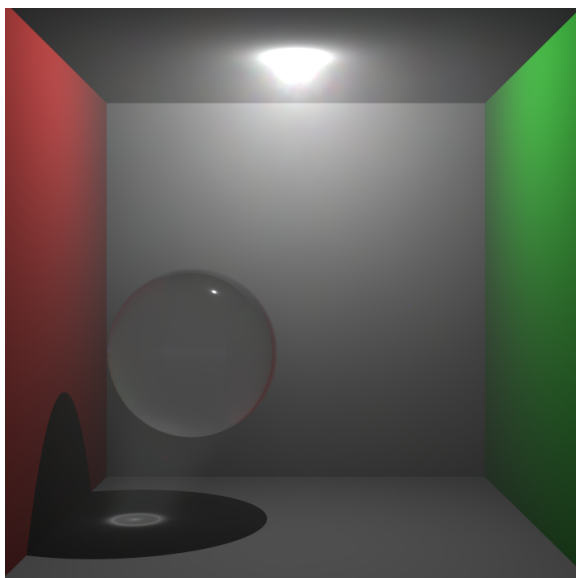
## 6.2 Isotropic Kernel vs Anisotropic Kernel

In this section it will be shown how the use of anisotropic kernel and photon differentials drastically improves the quality of the results.

Volumetric Photon Mapping [JC98] and the beam radiance estimate [JZJ08] make use of isotropic kernel functions, even though they are not using photon differentials to compute the bandwidth of each photon.

In the implementation provided in this project, an isotropic kernel function can be used by storing the photons as spheres instead of ellipsoids.

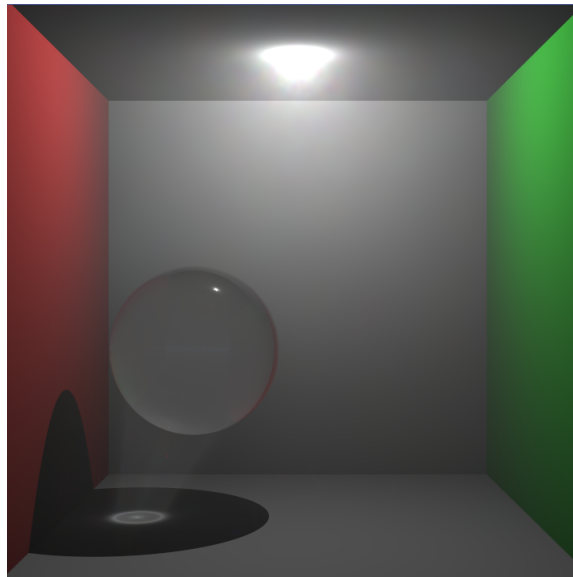
The results presented in this section were obtained by using the isotropic estimate (4.26) and the anisotropic estimate (4.30).



**Figure 6.7:** Cornell Box and glass ball inside a medium rendered with an isotropic kernel function.

The test scene used in Figure 6.7 and in Figure 6.8 contains a Cornell Box, a glass ball, and a point light and the whole scene is surrounded by a participating medium. Both the results were obtained by using approximately 5 million photons. The light coming from the point light is scattered inside the Cornell Box, and the light refracted by the glass ball generates a volume caustic, and a surface caustic on the floor of the box. The result in Figure 6.7 is obtained by using an isotropic kernel: the volume caustic is visible but the bias introduced by the density estimation blurs the edges of the caustic. An anisotropic ker-



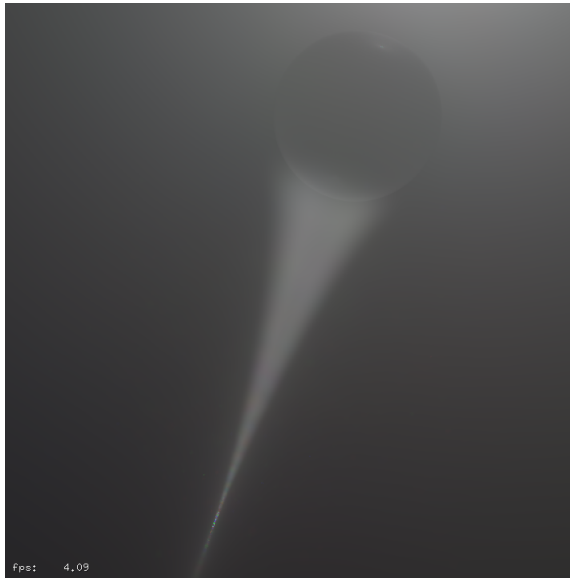


**Figure 6.8:** Cornell Box and glass ball inside a medium rendered with an anisotropic kernel function.

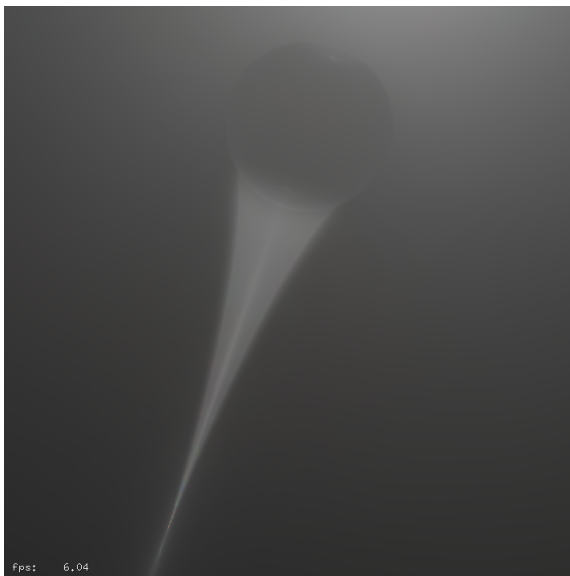
nel was used in Figure 6.8, and it can be seen how photon differentials reduce bias and noise, and the edges of the volume caustic are sharp and clearly visible.

Another interesting test scene is the one presented in Figure 6.9 and 6.10: a glass ball illuminated by a point light generates a volume caustic inside the medium; in this case 10 million photons have been used. The isotropic kernel, Figure 6.9, introduces blurring of the illumination features and it does not completely remove the noise (visible in the lower part of the volume caustic). On the other hand, in Figure 6.10 anisotropic filtering provides a very clear and detailed volume caustic.

In conclusion, photon differentials and anisotropic kernel functions provide a very efficient way to reduce bias and noise not only for surfaces, as shown in Chapter 2, but also for volumes.



**Figure 6.9:** Glass ball and point light in a black environment rendered with an isotropic kernel function.



**Figure 6.10:** Glass ball and point light in a black environment rendered with an anisotropic kernel function.

## 6.3 Analysis of the Radiance Estimate

The last comparison that will be made is between the estimate (3.19) of the RTE presented by Schjøth and the one developed in this thesis.

Three different test scenes have been used. The first scene consists of a glass ball inside a participating medium in a black environment, and a point light; for this scene approximately 10 million photons have been used. The second scene includes a Cornell Box, a point light, and the glass ball, the number of photons for this scene is around 5 million. The third and last scene consists of a diffuse plane, the glass ball, and a spot light, for a total of 3 million photons.

In the first test scene, the light coming from the point light is scattered around the scene, and the light that passes through the glass ball produces a volume caustic. In Figure 6.11 the result obtained by using equation (3.19) is shown, while in Figure 6.12 the estimate (4.30) is used. In Figure 6.13 and in Figure 6.14 two details of the results are shown. It is possible to notice how with almost the same number of photons the result in Figure 6.12 is less noisy and more bright than the one in Figure 6.11.

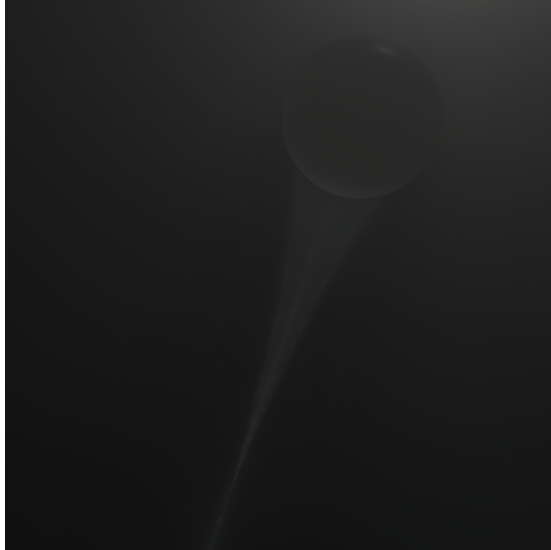
In the second test scene, the light coming from the point light source is refracted by the glass ball and produces a volume caustic, and a surface caustic. In Figure 6.15 and in Figure 6.16 the two different results are shown. In the details of Figure 6.17 and Figure 6.18 it is visible how the problems of the first test scene are present also in this case. The rendered image obtained with the estimate proposed by Schjøth is darker and more noisy than the one produced with the new radiance estimate.

In the third scene, a spot light produces a cone of light that illuminates the glass ball, which generates a volume caustic and a surface caustic on the diffuse plane. The glass ball blocks part of the light and produces a cone of shadow. Once again the result obtained with the radiance estimate presented in this thesis, Figure 6.20 and Figure 6.22, is brighter; in particular the difference is visible in the area of the cone of light close to the upper part of the sphere, and in the lower part of the volume caustic.

In conclusion, the main difference between the two methods is the brightness of the scattered light. The estimate (3.19) presented by Schjøth integrates the kernel function along the segment representing the overlap of a photon with the eye ray, but it does not consider that also the transmittance is a function of

the position and then it should be integrated together with the kernel function. What Schjøth does is to use the value of the transmittance calculated at the middle point of the segment as approximation of the integral. Since the transmittance function is exponential and not linear, if the length of the segment is long enough the value calculated at the starting point of the segment and the one computed at the ending point might be very different from the one computed at the middle point.

For this reason, the estimate used by Schjøth loses a small part of the radiance by approximating the value of the transmittance along the overlapped segment. The radiance estimate presented in this thesis does not have this problem, as it computes the exact contribution of the transmittance and of the kernel function.



**Figure 6.11:** Glass ball in black environment, Schjøth estimate.



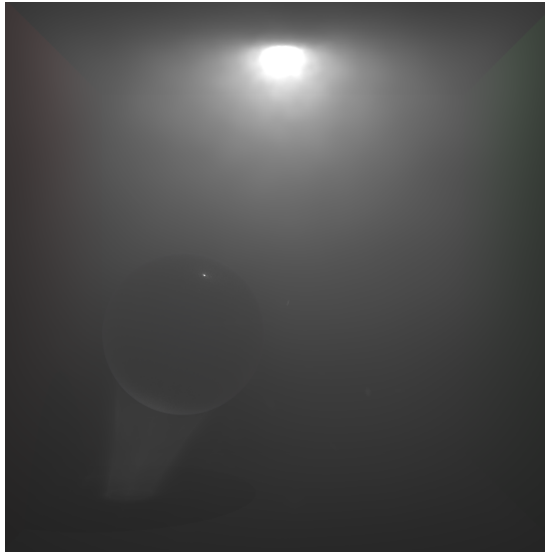
**Figure 6.12:** Glass ball in black environment, new developed estimate.



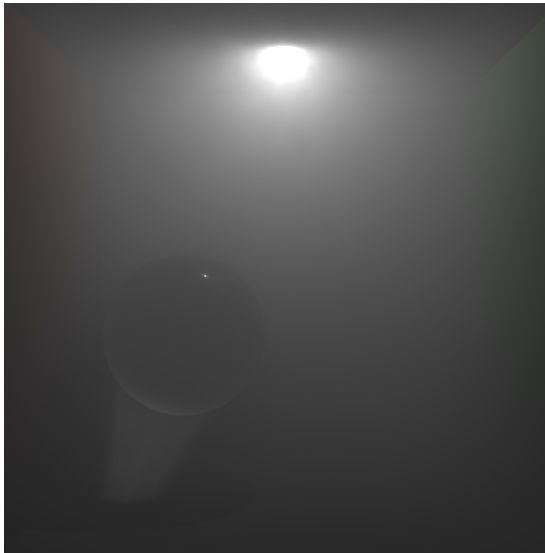
**Figure 6.13:** Glass ball in black environment, Schjøth estimate. (**Detail**)



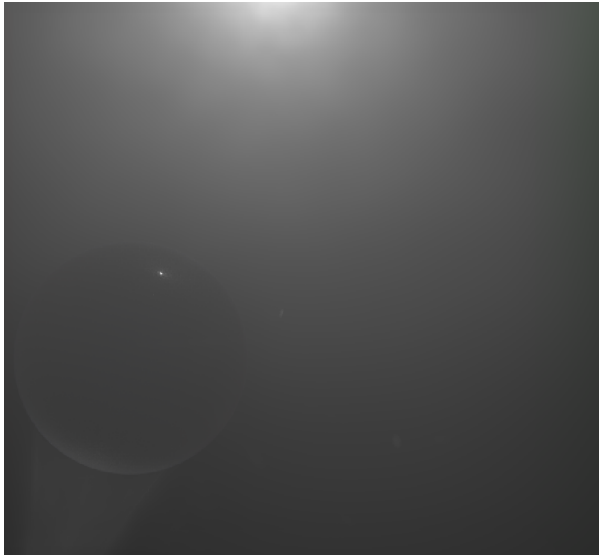
**Figure 6.14:** Glass ball in black environment, new developed estimate. (**Detail**)



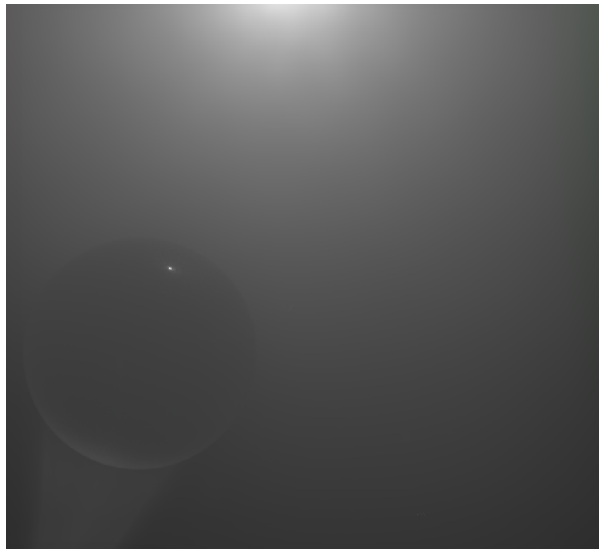
**Figure 6.15:** Cornell Box and glass ball inside a medium, Schjøth estimate.



**Figure 6.16:** Cornell Box and glass ball inside a medium, new developed estimate.

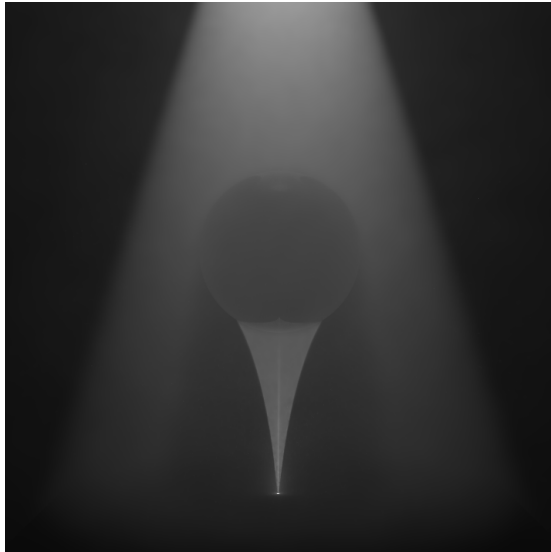


**Figure 6.17:** Cornell Box and glass ball inside a medium, Schjøth estimate. (Detail)

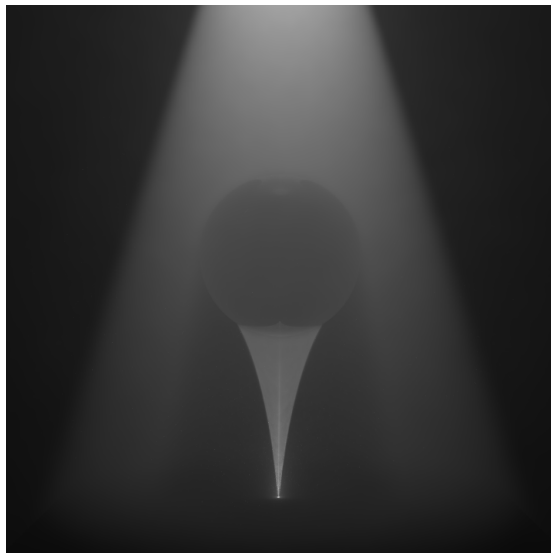


**Figure 6.18:** Cornell Box and glass ball inside a medium, new developed estimate. (Detail)

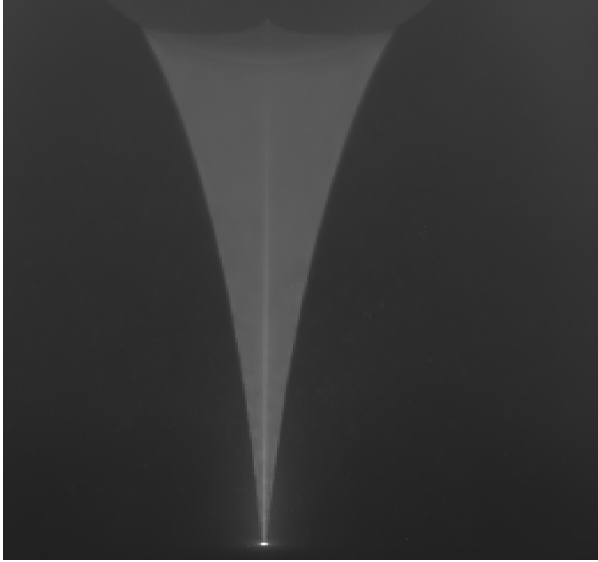




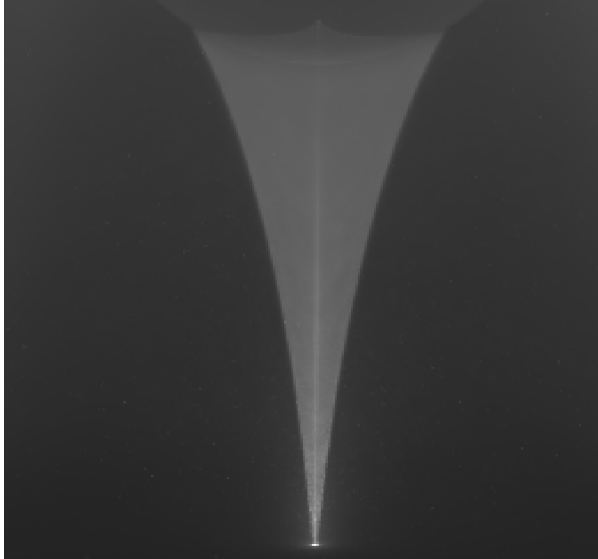
**Figure 6.19:** Glass ball and diffuse plane illuminated by a spot light, Schjøth estimate.



**Figure 6.20:** Glass ball and diffuse plane illuminated by a spot light, new developed estimate.



**Figure 6.21:** Glass ball and diffuse plane illuminated by a spot light, Schjøth estimate. (**Detail**)



**Figure 6.22:** Glass ball and diffuse plane illuminated by a spot light, new developed estimate. (**Detail**)

## CHAPTER 7

# Conclusions and Future Work

---

In this thesis it has been presented a description of how the light is affected by participating media and how photon differentials can be used for rendering volumes. The focus was on investigating and improving already existing algorithms that make use of photon differentials. The implemented algorithm has been developed by using the *NVIDIA*® *OptiX*<sup>TM</sup> ray tracing engine, and it supports a new estimate of the radiative transfer equation based on photon differentials.

In Chapter 2 it was introduced a global illumination algorithm based on Photon Mapping. It was shown how density estimation could be used to compute the light transport equation at the cost of introducing a trade-off between noise and bias. If few photons were used in the estimate then the results presented noise, on the other hand if too many photons were used then the illumination features resulted to be blurry. It was also shown how photon differentials and anisotropic filtering could be used to improve this trade-off and to reduce both noise and bias with no need of storing more photons.

In the first section of Chapter 3 it was presented the behaviour of the light when it is not travelling in a vacuum. The presence of a participating medium affects

the path of the light in different ways. Light might be emitted or absorbed by the particles that are inside the medium, a portion of the light might also be scattered in different directions, and all these phenomena can be described by the radiative transfer equation RTE. Some of the algorithms that have been developed over the years were presented in the second section of the chapter: Volumetric Photon Mapping, the beam radiance estimate, photon differentials, and photon beams.

In Chapter 4 it was described how to derive a more accurate estimate of the RTE than the one presented by Schjøth [Sch09]. In the first section it was described how to extend photon differentials from surfaces to volumes, and the sampling techniques for tracing photons that are used in this thesis were described. In the second section the focus was on the Ray Tracing pass, and the steps to compute the new radiance estimate were presented.

In Chapter 5 the three steps that have been followed in order to reach the final GPU implementation were described. The first step consisted in implementing a GPU version of photon mapping based on photon differential splatting. The implementation was inspired by the technique presented by Frisvad et al. [FSES14]. The second step was to implement a GPU algorithm for volume rendering by using photon differentials, and two different approaches have been tried: the first approach was the natural extension of the algorithm implemented during the first step, but it turned out to be inaccurate and inefficient; the second approach made a better use of the tools provided by OptiX and it uses the radiance estimate proposed by Schjøth [Sch09]. The last step was to research improvement for the algorithm, and the radiance estimate that was computed in Chapter 4 has been implemented into the code.

In Chapter 6 the results were shown. In the first section it was shown how absorption and scattering affect the propagation of the light and the appearance of the medium. In the second section the advantages of using anisotropic kernel functions instead of using isotropic kernel functions were shown. Finally a comparison between the results obtained with the radiance estimate presented in this thesis and the results obtained with the estimate presented by Schjøth [Sch09] has been made.

When this project was started it was known that five months would have not been enough to both implementing an algorithm for rendering volumes and researching new techniques to improve the existing algorithms. The work that has been presented represents a solid GPU algorithm that could be used as starting point from which it is possible to continue the research.

A possible extension for this thesis project might be the implementation of heterogeneous media, which has not been implemented due to lack of time. Another topic that might be interesting to look into is how to improve the scattering of photon differentials inside a participating medium, because the method used in this project is based on some heuristics presented by Schjøth [Sch09] and maybe a more accurate technique could be found.

One last suggestion for a future work is to implement progressive photon beams and to find improvements.

To sum up, this project represented a great opportunity to learn and understand volume rendering. It was not easy to reach the final results and it was also necessary to change the way of approaching the problems more than once, but it was worth it. A GPU algorithm for rendering volumes has been implemented and some improvements to the radiance estimate by using photon differentials have been found.

Hopefully the work presented in this thesis will provide inspiration for future research.



# APPENDIX A

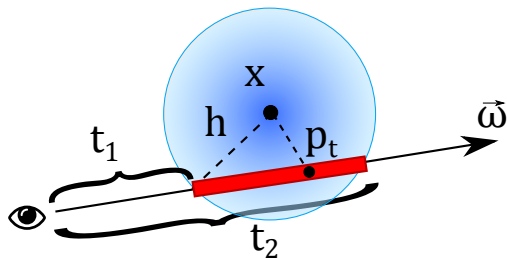
## Appendix A

---

In this section it will be shown that

$$L(\mathbf{p}, \vec{\omega}) \approx T_r(\mathbf{p}, \mathbf{p}_s)L(\mathbf{p}_s, \vec{\omega}) + \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_i} \frac{5}{2} \int_{t_{1,i}}^{t_{2,i}} e^{-\sigma_t t} \left( 1 - \left( \frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i} \right)^2 \right) dt, \quad (\text{A.1})$$

admits a solution.



PROOF. By calling  $\mathbf{p}_t = \mathbf{O} + t\vec{\omega}$ , where  $\mathbf{O}$  is the origin of the eye ray:

$$\|\mathbf{x} - \mathbf{p}_t\|^2 = \left(\frac{t_2 + t_1}{2} - t\right)^2 + h^2 - \left(\frac{t_2 - t_1}{2}\right)^2. \quad (\text{A.2})$$

By considering only the integral of equation (A.1) and equation (A.2):

$$\begin{aligned} S &= \int_{t_1}^{t_2} e^{-\sigma_t t} \left(1 - \left(\frac{\|\mathbf{x} - \mathbf{p}_t\|}{h}\right)^2\right) dt = \frac{1}{h^2} \int_{t_1}^{t_2} e^{-\sigma_t t} (-t^2 + t(t_1 + t_2) - t_1 t_2) dt \\ &= \frac{1}{h^2} \left[ - \int_{t_1}^{t_2} e^{-\sigma_t t} t^2 dt + \int_{t_1}^{t_2} e^{-\sigma_t t} t(t_2 + t_1) dt - \int_{t_1}^{t_2} e^{-\sigma_t t} t_1 t_2 dt \right] \\ &= \frac{1}{h^2} [-A + B - C]; \end{aligned} \quad (\text{A.3})$$

by integrating by parts  $A$ :

$$\begin{aligned} A &= \int_{t_1}^{t_2} e^{-\sigma_t t} t^2 dt = - \frac{e^{-\sigma_t t}}{\sigma_t} t^2 \Big|_{t_1}^{t_2} + \int_{t_1}^{t_2} \frac{e^{-\sigma_t t}}{\sigma_t} 2t dt \\ &= - \frac{e^{-\sigma_t t}}{\sigma_t} t^2 \Big|_{t_1}^{t_2} - \frac{e^{-\sigma_t t}}{\sigma_t^2} 2t \Big|_{t_1}^{t_2} + \int_{t_1}^{t_2} \frac{e^{-\sigma_t t}}{\sigma_t^2} 2 dt \\ &= - \frac{e^{-\sigma_t t}}{\sigma_t} t^2 \Big|_{t_1}^{t_2} - \frac{e^{-\sigma_t t}}{\sigma_t^2} 2t \Big|_{t_1}^{t_2} - \frac{e^{-\sigma_t t}}{\sigma_t^3} 2 \Big|_{t_1}^{t_2} \\ &= - \frac{e^{-\sigma_t t_2}}{\sigma_t} t_2^2 + \frac{e^{-\sigma_t t_1}}{\sigma_t} t_1^2 - \frac{e^{-\sigma_t t_2}}{\sigma_t^2} 2t_2 + \frac{e^{-\sigma_t t_1}}{\sigma_t^2} 2t_1 - \frac{e^{-\sigma_t t_2}}{\sigma_t^3} 2 + \frac{e^{-\sigma_t t_1}}{\sigma_t^3} 2. \end{aligned} \quad (\text{A.4})$$

By integrating by parts  $B$ :

$$\begin{aligned} B &= \int_{t_1}^{t_2} e^{-\sigma_t t} t(t_2 + t_1) dt = - \frac{e^{-\sigma_t t}}{\sigma_t} t(t_2 + t_1) \Big|_{t_1}^{t_2} + \int_{t_1}^{t_2} \frac{e^{-\sigma_t t}}{\sigma_t} (t_2 + t_1) dt \\ &= (t_1 + t_2) \left[ - \frac{e^{-\sigma_t t}}{\sigma_t} t \Big|_{t_1}^{t_2} - \frac{e^{-\sigma_t t}}{\sigma_t^2} \Big|_{t_1}^{t_2} \right] \\ &= (t_1 + t_2) \left[ - \frac{e^{-\sigma_t t_2}}{\sigma_t} t_2 + \frac{e^{-\sigma_t t_1}}{\sigma_t} t_1 - \frac{e^{-\sigma_t t_2}}{\sigma_t^2} + \frac{e^{-\sigma_t t_1}}{\sigma_t^2} \right]. \end{aligned} \quad (\text{A.5})$$

By integrating  $C$ :

$$C = \int_{t_1}^{t_2} e^{-\sigma_t t} t_1 t_2 dt = - \frac{e^{-\sigma_t t}}{\sigma_t} t_2 t_1 \Big|_{t_1}^{t_2} = - \frac{e^{-\sigma_t t_2}}{\sigma_t} t_2 t_1 + \frac{e^{-\sigma_t t_1}}{\sigma_t} t_2 t_1. \quad (\text{A.6})$$



By substituting equation (A.4), (A.5) and (A.6) into (A.3):

$$\begin{aligned}
S &= \frac{1}{\hbar^2} [-A + B - C] = \frac{1}{\hbar^2} \left[ + \frac{e^{-\sigma_t t_2}}{\sigma_t} t_2^2 - \frac{e^{-\sigma_t t_1}}{\sigma_t} t_1^2 + \frac{e^{-\sigma_t t_2}}{\sigma_t^2} 2t_2 - \frac{e^{-\sigma_t t_1}}{\sigma_t^2} 2t_1 \right. \\
&\quad + \frac{e^{-\sigma_t t_2}}{\sigma_t^3} 2 - \frac{e^{-\sigma_t t_1}}{\sigma_t^3} 2 + (t_1 + t_2) \left( -\frac{e^{-\sigma_t t_2}}{\sigma_t} t_2 + \frac{e^{-\sigma_t t_1}}{\sigma_t} t_1 - \frac{e^{-\sigma_t t_2}}{\sigma_t^2} + \frac{e^{-\sigma_t t_1}}{\sigma_t^2} \right) \\
&\quad \left. + \frac{e^{-\sigma_t t_2}}{\sigma_t} t_2 t_1 - \frac{e^{-\sigma_t t_1}}{\sigma_t} t_2 t_1 \right] \\
&= \frac{1}{\hbar^2 \sigma_t^2} \left[ (t_2 - t_1) (e^{-\sigma_t t_2} + e^{-\sigma_t t_1}) - \frac{2}{\sigma_t} (e^{-\sigma_t t_1} - e^{-\sigma_t t_2}) \right].
\end{aligned} \tag{A.7}$$

By replacing equation (A.7) into (A.1):

$$\begin{aligned}
L(\mathbf{p}, \vec{\omega}) &\approx T_r(\mathbf{p}, \mathbf{p}_s) L(\mathbf{p}_s, \vec{\omega}) \\
&\quad + \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_i} \frac{5}{2} \int_{t_{1,i}}^{t_{2,i}} e^{-\sigma_t t} \left( 1 - \left( \frac{\|\mathbf{x}_i - \mathbf{p}_t\|}{h_i} \right)^2 \right) dt \\
&\approx T_r(\mathbf{p}, \mathbf{p}_s) L(\mathbf{p}_s, \vec{\omega}) \\
&\quad + \sum_{i=1}^N p(\mathbf{p}, \vec{\omega}'_i \rightarrow \vec{\omega}) \frac{\Delta\Phi_i}{V_i} \frac{5}{2} \frac{1}{h_i^2 \sigma_t^2} \\
&\quad \cdot \left[ (t_{2,i} - t_{1,i}) (e^{-\sigma_t t_{2,i}} + e^{-\sigma_t t_{1,i}}) - \frac{2}{\sigma_t} (e^{-\sigma_t t_{1,i}} - e^{-\sigma_t t_{2,i}}) \right].
\end{aligned} \tag{A.8}$$



# Bibliography

---

- [Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. of the ACM* 18, pages 509–517, 1975.
- [Cha60] Subrahmanyan Chandrasekhar. Radiative transfer. *Dover Publications*, 1960.
- [ConTCa] Wikipedia Contributors. Emission spectrum. *Wikipedia, The Free Encyclopedia*, 26 May 2015 14:15 UTC. [http://en.wikipedia.org/w/index.php?title=Emission\\_spectrum&oldid=648368936](http://en.wikipedia.org/w/index.php?title=Emission_spectrum&oldid=648368936).
- [ConTCb] Wikipedia Contributors. Absorption (electromagnetic radiation). *Wikipedia, The Free Encyclopedia*, 27 May 2015 07:52 UTC. [http://en.wikipedia.org/w/index.php?title=Absorption\\_\(electromagnetic\\_radiation\)&oldid=660606811](http://en.wikipedia.org/w/index.php?title=Absorption_(electromagnetic_radiation)&oldid=660606811).
- [ConTCc] Wikipedia Contributors. Quaternion. *Wikipedia, The Free Encyclopedia*, 31 May 2015 15:14 UTC. <http://en.wikipedia.org/w/index.php?title=Quaternion&oldid=662856353>.
- [CorTC] NVIDIA Corporation. Optix documentation. 05 June 2015 22:15 UTC. <https://developer.nvidia.com/optix-documentation>.
- [FSES14] J. R. Frisvad, L. Schøth, K. Erleben, and J. Sporring. Photon differential splatting for rendering caustics. *Computer Graphics forum vol.33*, pages 252–263, 2014.
- [Fur14] Ian Furst. Tree in field during extreme cold with frozen fog. 2014. Licensed under CC BY-SA 3.0, [http://en.wikipedia.org/wiki/Fog#/media/File:Tree\\_in\\_field\\_during\\_extreme\\_cold\\_with\\_frozen\\_fog.png](http://en.wikipedia.org/wiki/Fog#/media/File:Tree_in_field_during_extreme_cold_with_frozen_fog.png).

- [Hec90] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *SIGGRAPH Comput. Graph*, pages 145–154, 1990.
- [Ige99] H. Igehy. Tracing ray differentials. *Computer Graphics Proceedings SIGGRAPH 1999*, 1999.
- [JC95] H. W. Jensen and N. J. Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computer and Graphics*, 1995.
- [JC98] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. *SIGGRAPH*, pages 311–320, 1998.
- [Jen96] H. W. Jensen. *The Photon Map in Global Illumination*. PhD thesis, Technical University of Denmark, 1996.
- [JNSJ11] W. Jarosz, D. Nowrouzezahrai, I. Sadeghi, and H. W. Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions of Graphics (Presented at SIGGRAPH 2011)*, 2011.
- [JNT<sup>+</sup>11] W. Jarosz, D. Nowrouzezahrai, R. Thomas, P.P. Sloan, and M. Zwicker. Progressive photon beams. *ACM Transaction of Graphics (Presented at SIGGRAPH 2011)*, 2011.
- [JZJ08] W. Jarosz, M. Zwicker, and H. Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, pages 557–556, 2008.
- [Kaj86] J. T. Kajiya. The rendering equation. *Computer Graphics 20*, pages 143–150, August 1986.
- [Lac06] Steve Lacy. Morning mist. 2006. Licensed under CC BY-NC-ND 2.0, [https://www.flickr.com/photos/steve\\_lacy/142536262/](https://www.flickr.com/photos/steve_lacy/142536262/).
- [LW93] E. P. Lafortune and D.Y. Willems. Bi-directional path tracing. *Compugraphics*, pages 145–154, 1993.
- [LW96] E. P. Lafortune and D.Y. Willems. Rendering participating media with bidirectional path tracing. *EG Rendering Workshop*, pages 91–100, 1996.
- [MP00] A. Keller M. Pauly, T. Kollig. Metropolis light transport for participating media. *11th EG Workshop on Rendering*, pages 11–22, 2000.

- [Mys97] K. Myszkowski. Lighting reconstruction using fast and adaptive density estimation techniques. *Proceedings of the Eurographics Workshop on Rendering Technique*, pages 251–262, 1997.
- [Ols11] Frank Olsen. Red and green auroras, norway. 2011. Licensed under CC BY-SA 3.0, [http://en.wikipedia.org/wiki/Aurora#/media/File:Red\\_and\\_green\\_auroras.jpg](http://en.wikipedia.org/wiki/Aurora#/media/File:Red_and_green_auroras.jpg).
- [Ott06] Heiner Otterstedt. Caustics produced by a glass of water. 2006. Licensed under CC BY-SA 3.0, [http://en.wikipedia.org/wiki/Caustic\\_%28optics%29#/media/File:Kaustik.jpg](http://en.wikipedia.org/wiki/Caustic_%28optics%29#/media/File:Kaustik.jpg).
- [PBD<sup>+</sup>10] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 2010.
- [PM93] S. N. Pattanaik and S. P. Mudur. Computation of global illumination in a participating medium by monte carlo simulation. *The Journal of Visualization and Computer Animation*, 1993.
- [Sch03] R. Schregle. Bias compensation for photon maps. *Computer Graphics Forum*, pages 729–742, 2003.
- [Sch09] L. Schjøth. *Anisotropic Density Estimation in Global Illumination: a journey through time and space*. PhD thesis, University of Copenhagen, 2009.
- [SFES07] L. Schjøth, J. R. Frisvad, K. Erleben, and J. Sporring. Photon differentials. *Proceedings of GRAPHITE*, pages 179–186, 2007.
- [SH02] R. Siegel and J. R. Howell. Therma radiation heat transfer. *fourth ed. Taylor & Francis*, 2002.
- [VG94] E. Veach and L. Guibas. Bidirectional estimators for light transport. *Fifth Eurographics Workshop on Rendering*, pages 147–162, 1994.
- [Wal98] B. Walter. *Density estimation techniques for global illumination*. PhD thesis, Cornell University, 1998.