Legible Visualization of Semi-Transparent Objects using Light Transport

David Murray

LP2N

université de BORDEAUX

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **David Murray**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Visualisation d'objets semi-transparents basée sur le transport lumineux

---

Date de soutenance : 10 décembre 2018

Devant la commission d'examen composée de :

Xavier Granier ...	Professeur, Institut d'Optique	Directeur
Jérôme Baril	Ingénieur, Thermo Fisher Scientific	Co-directeur
Pierre Poulin	Professeur, DIRO, Université de Montréal	Président du jury
Mathias Paulin	Professeur, IRIT, Université de Toulouse .	Rapporteur
Daniel Meneveaux	Professeur, XLIM, Université de Poitiers .	Rapporteur

– 2018 –

Titre Visualisation d'objets semi-transparents basée sur le transport lumineux

Résumé

Explorer et comprendre des données volumétriques ou surfaciques est un des nombreux enjeux du domaine de l'informatique graphique. L'apparence de telles données peut être modélisée et visualisée en utilisant la théorie du transport lumineux. Afin de rendre une telle visualisation compréhensible, le recours à des matériaux transparents est très répandu. Si des solutions existent pour simuler correctement la propagation de la lumière et ainsi afficher des objets semi-transparents, offrir une visualisation compréhensible reste un sujet de recherche ouvert. Le but de cette thèse est double. Tout d'abord, une analyse approfondie du modèle optique pour le transport de la lumière et ses implications sur la génération d'images par ordinateur doit être effectuée. Ensuite, cette connaissance pourra être utilisée pour proposer des solutions efficaces et fiables pour visualiser des milieux transparents et semi-transparents.

Dans ce manuscrit, premièrement, nous présentons le modèle optique communément utilisé pour modéliser le transport de la lumière dans des milieux participatifs, sa simplification si l'on réduit la situation à des surfaces et la manière dont ces modèles sont utilisés en informatique graphique pour générer des images.

Deuxièmement, nous présentons une solution pour améliorer la représentation des formes dans le cas particulier des surfaces. La technique proposée utilise le transport lumineux comme base pour modifier le processus d'éclairage et modifier l'apparence et l'opacité des matériaux.

Troisièmement, nous nous concentrons sur la problématique de l'utilisation de données volumétriques au lieu du cas simplifié des surfaces. Dans ce cas, le fait de ne modifier que les propriétés du matériau a un impact limité. Nous étudions donc comment le transport lumineux peut être utilisé pour fournir des informations utiles à la compréhension de milieux participatifs.

Enfin, nous présentons notre modèle de transport lumineux pour les milieux participatifs, qui vise à explorer une région d'intérêt d'un volume.

Mots-clés Rendu, Carte Graphique, Rasterisation, Temps-Réel, Eclairage Global, Rendu Volumique, Diffusion

Title Legible Visualization of Semi-Transparent Objects using Light Transport

Abstract

Exploring and understanding volumetric or surface data is one of the challenges of Computer Graphics. The appearance of these data can be modeled and visualized using light transport theory. For the sake of understanding such a data visualization, transparent materials are widely used. If solutions exist to correctly simulate light propagation and display semi-transparent objects, offering an understandable visualization remains an open research topic. The goal of this thesis is twofold. First, an in-depth analysis of the optical model for light transport and its implication on computer generated images is performed. Second, this knowledge can be used to tackle the problematic of providing efficient and reliable solution to visualize transparent and semi-transparent media.

In this manuscript, we first introduce the general optical model for light transport in participating media, its simplification to surfaces, and how it is used in computer graphics to generate images.

Second, we present a solution to improve shape depiction in the special case of surfaces. The proposed technique uses light transport as a basis to change the lighting process and modify material appearance and opacity.

Third, we focus on the problem of using full volumetric data instead of the simplified case of surfaces. In this case, changing only material properties has a limited impact, thus we study how light transport can be used to provide useful information for participating media.

Last, we present our light transport model for participating media that aims at exploring part of interest of a volume.

Keywords Rendering, GPU, Rasterization, Real-Time, Global Illumination, Volume Rendering, Scattering

Laboratoire d'accueil Laboratoire Photonique, Numérique, Nanosciences LP2N (UMR 5298) CNRS - Thermo Fisher Scientific R&D

Remerciements

Je tiens tout d'abord à remercier mes encadrants de thèse, Xavier Granier et Jérôme Baril, sans qui les travaux réalisés durant cette thèse n'auraient jamais aboutis. Je remercie Xavier, pour tout ce qu'il fait à l'Institut d'Optique à Bordeaux, où est né mon intérêt pour la synthèse d'image. Sans son enthousiasme et sa rigueur à enseigner, je n'aurai sans doute jamais eu l'occasion d'effectuer ma thèse dans ce domaine. Je le remercie ensuite cette fois en tant que directeur de ma thèse, pour toutes les discussions constructives que nous avons pu avoir, pour ses conseils, ainsi que pour son soutien pendant ces trois années. Je remercie également Jérôme. Cette aventure aura commencé par un petit stage pour finalement aboutir ici, et cela n'aurait pas été possible sans son soutien. Je le remercie donc pour son exigence scientifique, sa rigueur et pour toutes les discussions que j'ai eu avec lui, que ce soit pour avoir des explications techniques, pour parler de Coccinelle ou de coupe du monde.

Je remercie les deux rapporteurs de mon manuscrit de thèse, Daniel Meneveaux et Mathias Paulin, pour l'intérêt qu'ils ont porté aux travaux présentés ainsi que pour leur remarques pertinentes et constructives. Je remercie également Pierre Poulin, qui m'a fait l'honneur de présider mon jury de thèse. Je le remercie également pour le temps qu'il a pu m'accorder lors de son séjour à Bordeaux et pour nos discussions qui ont toujours été un moment que j'appréciais tant pour ses conseils avisés que pour l'humour dont il sait faire preuve. Je remercie également Thierry Dufour et Thermo Fisher Scientific pour avoir financé mon doctorat.

Enfin, beaucoup de personnes m'ont aidé, encouragé et supporté pendant ces travaux. Tout d'abord, je remercie mes collègues de Thermo Fisher Scientific, pour l'aide qu'ils m'ont apporté avec Open Inventor mais surtout pour les moments de détentes sportives, de célébration et les diverses discussions autour d'un café. Merci également à mes collègues de MANAO pour les nombreuses discussions scientifiques et celles qui l'étaient parfois beaucoup moins.

D'un point de vue plus personnel, je remercie mes amis, pour tous les bon moments passés ensemble. Une mention spéciale va à Sebastien, qui a aussi eu la lourde tache de relire mon manuscrit. Je remercie aussi ma famille, et plus particulièrement mes parents et mes grand-parents, qui m'ont supporté dans les différentes étapes de ma vie jusqu'ici et qui ont fait naitre tôt chez moi un attrait pour la science et la recherche.

Enfin, je remercie Maëlig, ma compagne, pour sa patience et sa confiance. Ces trois années aurait sans doute été beaucoup plus délicates sans son soutien, et sa présence au quotidien m'a beaucoup aidé à mener jusqu'au bout cette aventure qu'est la thèse.

Résumé long

Contexte et motivations

L'informatique graphique est la science qui inclut toutes les méthodes pour transmettre des informations visuelles à l'aide d'un ordinateur. En particulier, la synthèse (ou le rendu) d'images permet la production d'images à l'aide d'une scène numérique. Il s'agit d'un processus répandu, utilisé dans de nombreux domaines: films d'animation par ordinateur , effets spéciaux, jeux vidéo, visualisation scientifique ...

Cette thèse est menée au sein de Thermo Fisher Scientific™ pour le développement de démonstrateurs pour sa plateforme de logiciel Open Inventor®. Son objectif principal est de fournir des solutions pour la visualisation de données scientifiques. Parmi les domaines d'utilisation de ces données, on peut citer notamment : le domaine médical, l'analyse sismique et l'analyse des sols, les sciences naturelles, la conception et l'inspection industrielles, etc.

La plupart de ces domaines recourt régulièrement à l'utilisation d'objets semi-transparents (définis comme des couches de surfaces transparentes ou des volumes continus) pour l'exploration et la visualisation d'objets complexes. Afficher correctement des objets transparents nécessite de comprendre comment la lumière est censée interagir avec ces objets, mais aussi de transmettre ces informations de manière interactive et lisible. Le premier problème peut se résoudre avec la théorie du transport de la lumière, le second en choisissant la méthode de rendu appropriée.

La méthode de rendu dépend fortement du besoin de l'application et, par conséquent, le style souhaité n'est pas nécessairement le même. Plusieurs méthodes de rendu ont été conçues pour répondre à cette diversité. Par exemple, le rendu physiquement plausible (Lewis [1993]) consiste à décrire aussi fidèlement que possible les effets produits par l'interaction lumière-matière (également appelée transport de lumière, Veach and Guibas [1997]). A l'opposé, le rendu expressif (Lansdown and Schofield [1995]), également appelé rendu non photoréaliste, détourne et modifie le transport de lumière à des fins de stylisation, e.g., le style illustratif (Gooch and Gooch [2001]).

Dans cette thèse, nous nous intéressons donc à ces objets semi-transparents (les surfaces transparentes et, principalement, les volumes) pour en améliorer

la perception. Nous proposons d'abord une solution pour améliorer la lisibilité des surfaces transparentes. Nous introduisons ensuite une nouvelle approche utilisant le transport lumineux pour explorer des volumes.

Organisation du document

Ce manuscrit est divisé en quatre chapitres.

Le chapitre 1 présente les différentes bases scientifiques et techniques nécessaires pour comprendre les concepts développés dans le manuscrit, ainsi que les problèmes que nous souhaitons résoudre. À cette fin, nous discutons du modèle d'interaction lumière-matière appliqué aux volumes et aux surfaces, ainsi que de son utilisation en informatique graphique. Nous présentons ensuite quelques techniques de rendu expressif. Enfin, nous présentons également Open Inventor, la boîte à outils de Thermo Fisher Scientific ™, dans laquelle nous devons développer les solutions présentées dans ce manuscrit.

Le chapitre 2 cible les problèmes spécifiques des surfaces transparentes et les solutions pour le rendu de celles-ci. Le principal problème consiste à offrir une visualisation lisible de différentes couches de surfaces transparentes. Nous proposons une solution visant à améliorer la représentation du relief de ces surfaces. En particulier, nous présentons dans ce chapitre plusieurs structures de données (y compris la nôtre) adaptées au rendu en temps réel de surfaces transparentes.

Le chapitre 3 se concentre sur les volumes continus. Les techniques présentées dans le chapitre 2 étant limitées lorsqu'elles sont appliquées à des volumes, nous introduisons une nouvelle approche inspirée de la fluoroscopie en médecine. Cette nouvelle métaphore est basée sur le transport lumineux pour améliorer la représentation des volumes. Nous étudions également différentes approches numériques afin de déterminer les caractéristiques les mieux adaptées à notre application.

Basé sur ces caractéristiques, le Chapitre 4 présente la technique de rendu que nous avons choisie pour résoudre le transport de la lumière dans des volumes à des fins d'exploration expressive. En effet, nous expliquons comment utiliser ce modèle pour identifier une région d'intérêt. Nous présentons également quelques outils et pistes de réflexion pour utiliser ce modèle.

Conclusion

Dans ce document, nous nous sommes tout d'abord intéressés au transport de la lumière et à son implication dans le rendu de surfaces et de volumes transparents. Nous avons également étudié comment le modifier pour fournir des informations supplémentaires à l'aide de techniques de rendu non photoréalistes. En particulier, nous avons comparé les techniques utilisées dans la

littérature avec les fonctionnalités d'Open Inventor.

Les conclusions de cette étude ont conduit au développement d'une solution pour améliorer la représentation de forme pour les surfaces transparentes. Notre proposition concernait deux aspects du processus de rendu. Premièrement, pour calculer efficacement des gradients d'informations dans un espace 3D, nous concevons une structure de données permettant un calcul rapide de ces informations. Deuxièmement, nous avons proposé de moduler l'opacité des surfaces à l'aide d'informations géométriques telles que la courbure de surface. Cette partie du travail a été publiée lors de la conférence EuroGraphics Symposium Rendering en 2016 (Murray *et al.* [2016]).

Après avoir abordé le cas des surfaces transparentes, nous avons abordé le cas des volumes. Nous avons d'abord étudié la manière dont notre solution précédente pouvait s'étendre aux volumes, qui s'est avéré n'être adapté qu'aux isosurfaces. En retournant aux bases du transport lumineux (l'équation de transfert radiatif, ETR), nous avons essayé de le modifier dans le but de pouvoir contraindre la diffusion de la lumière. Nous avons d'abord testé l'idée, avec succès, avec une version basée sur l'algorithme de Diffusion Anisotropique (Perona and Malik [1990]). Nous avons ensuite étudié différentes techniques numériques afin de déterminer celle qui convient le mieux pour obtenir le même type de résultats avec l'ETR. Nous avons tiré deux conclusions importantes de cette étude. Premièrement, la résolution itérative de l'ETR, dans sa forme stationnaire, à l'aide d'une méthode de Jacobi n'est pas stable. Deuxièmement, résoudre l'ETR instationnaire de manière itérative avec une base directionnelle conduit à des artefacts sous forme de grille.

Enfin, en utilisant l'approximation de diffusion (Stam [1995]), associée à l'hypothèse d'un flux constant, nous avons réduit le problème à un modèle ayant la forme d'une équation de diffusion. Ce modèle ne présente pas les mêmes artefacts que celui observé lors de notre précédente étude, tout en étant stable pour une gamme de paramètres satisfaisante. Nous l'avons ensuite utilisé pour effectuer une diffusion sélective en modifiant les paramètres de l'ETR pour atteindre notre objectif. Certains outils ont également été proposés pour interagir avec le processus de diffusion. Cependant, au moment de la rédaction de ce document, ce travail est toujours en développement, particulièrement en ce qui concerne la visualisation et l'interaction.

Concernant les travaux spécifiques à Open Inventor, la thèse a conduit au développement de deux démonstrateurs: un pour le rendu expressif sur les surfaces (opaques et transparentes) et l'autre pour le calcul de la visibilité dans les volumes (basé sur la technique de Jönsson *et al.* [2012]). Un dernier est en cours de développement pour démontrer les possibilités de la diffusion sélective.

Travaux futurs

Surfaces

Vers l'illumination globale Nous avons présenté une structure de données permettant d'accéder efficacement aux voisins dans une représentation à plusieurs couches. Nous pensons que l'intérêt de cette structure ne se limite pas à la représentation des formes. En effet, nous pourrions augmenter la quantité de phénomènes optiques que nous simulons actuellement. Cette structure pourrait être utilisée pour effectuer une intersection approximative de rayons et ainsi simuler une réflexion ou une réfraction. De plus, un accès efficace au voisinage réel, ainsi que des informations sur la profondeur, pourraient être utilisés pour estimer la diffusion opérant sous la surface, ou la translucidité, tout en ayant un impact limité sur les performances.

Dérivées du troisième ordre Dans notre solution, comme nous nous concentrions sur la modulation d'opacité, les seules fonctionnalités de rendu de ligne actuellement prises en charge sont les contours occlusifs. Ainsi, nous prévoyons d'étudier l'impact de l'utilisation du rendu par lignes, basé sur les points d'inflexion (DeCarlo *et al.* [2003]; Ohtake *et al.* [2004]; Judd *et al.* [2007]; Kolomenkin *et al.* [2008]), afin de mettre en valeur ce type d'information sur des surfaces transparentes.

Il est à noter que l'utilisation de fonctionnalités de troisième ordre nécessite également d'effectuer une autre passe de dérivation, ce qui peut s'avérer coûteux pour être réalisé en temps réel. L'extension à des fonctionnalités de troisième ordre peut alors nécessiter d'adapter notre structure de données afin de stocker plus d'informations et d'éviter un surcout en calcul important.

Support pour une étude utilisateur Les résultats présentés dans le chapitre 2 sont principalement basés sur des retours de collègues et proches collaborateurs. Notre conclusion sur l'impact de ces travaux est probablement biaisée.

Cependant, une étude utilisateur permettrait de déterminer plus efficacement l'impact de notre solution sur la perception de la forme de l'objet. En particulier, si le rendu par lignes est ajouté, il serait intéressant d'en étudier l'impact sur la lisibilité.

Volumes

Optimisations Comme indiqué au chapitre 4, l'application de diffusion sélective doit être optimisée. Un aspect que nous n'avons pas encore pris en compte consiste à utiliser une résolution plus petite pour évaluer la diffusion. Cela permettrait à la fois de réduire l'utilisation en mémoire et les coûts de calcul. Cependant, cela implique que les résultats seront probablement moins précis.

En outre, une autre solution consisterait à réduire la précision de l'information que nous traitons, mais cela présente le même problème que celui de la diminution de la résolution. Nous devons donc mener une étude pour déterminer un bon équilibre entre précision et efficacité. Cette étape est cruciale si nous voulons fournir une solution efficace et par la suite intégrer cette application à Open Inventor.

En outre, en fonction de la conclusion de cette étude, il pourrait être intéressant d'exploiter niveaux de détail des textures. En utilisant correctement les différents niveaux, nous pourrions adapter le noyau de calcul pour fournir des résultats plus rapides, mais moins précis. Cependant, cela induira probablement une surcharge de mémoire par rapport à l'utilisation du niveau supérieur uniquement.

L'augmentation de la vitesse de convergence améliorera l'interactivité de nos techniques. Cet aspect, associé à une mémoire réduite, devrait encourager l'utilisation d'une telle approche.

Visualisation Comme indiqué au chapitre 4, l'algorithme effectue une diffusion sélective, mais la visualisation de ses résultats reste limitée. La visualisation étant un élément important de l'application, elle doit être traitée à l'avenir. Cela nécessitera probablement une interaction plus étroite avec nos ingénieurs d'application et nos spécialistes en interfaces graphiques.

En outre, nous devrions étudier quelles grandeurs (e.g., taux de fluence, luminance, gradients, etc) sont importantes à afficher pour la compréhension du processus. Nous pouvons également étudier la manière dont des fonctions de transfert bien définies pourraient être utiles (Ljung *et al.* [2016]) pour cette question, car cet aspect n'était pas au centre de cette thèse.

Interactions Nous avons introduit de nombreux outils pour manipuler l'algorithme. Cependant, certains d'entre eux peuvent être améliorés pour être plus intuitifs. En particulier, nous envisageons d'ajouter la possibilité d'interagir avec la configuration d'éclairage en sélectionnant directement sur l'écran (sur les tranches du volumes) les positions des sources lumineuses. Cela rendrait également l'outil de peinture plus facile à manipuler et pourrait permettre d'explorer des configurations plus élaborées.

Nous devrions également proposer un moyen intuitif de choisir les différentes fonctions de transfert utilisées, autant pour le calcul que l'affichage. Dans le même but, nous devons veiller à ce que la manière dont les paramètres seront exposés reste intuitive pour l'utilisateur. Nous devons trouver une solution pour que l'interaction avec les paramètres soit intuitive, tout en s'assurant de rester dans le domaine de convergence de la méthode.

Vers l'illumination globale En raison des diverses approximations que nous avons effectuées, notre méthode est limitée quant au type de milieux et

de matériaux qu'elle peut traiter. Ainsi, notre solution ne peut pas évaluer l'illumination globale aussi précisément que les techniques stochastiques.

Cependant, son principal avantage est sa capacité à se corriger automatiquement. Grâce à cela, l'algorithme peut supporter de nombreuses modifications en terme de paramètres et de configuration de lumière (position, intensité...). Ainsi, il pourrait être utilisé pour permettre de trouver une bonne configuration d'éclairage lors du rendu des volumes.

En outre, il pourrait être utilisé pour le rendu de surface pure afin de réaliser une diffusion sous-surface. En effet, la diffusion sous-surface est utilisée pour approximer le transport de la lumière dans les milieux participants lors du rendu de surfaces pures. En utilisant des petits volumes répartis sur les surfaces, notre algorithme pourrait être utilisé pour évaluer cette diffusion.

Contents

List of Figures

Notation and Acronyms

Notations

p	spatial position.
$\vec{\omega}$	vector.
$u^T \cdot v$	scalar product between two vectors.
$\vec{\nabla}$	gradient operator.
$div(\vec{\omega})$	divergence of a vector, analogous to $\vec{\nabla}^T \cdot \vec{\omega}$.
$\vec{\omega_i}$	incident direction (toward light), normalized.
$\vec{\omega_o}$	outgoing direction (toward eye), normalized.
$\vec{n}$	normal vector of a surface, normalized.
Ω	unit hemisphere integration domain.
Ω^2	unit sphere integration domain.
L	radiance.
E	irradiance.

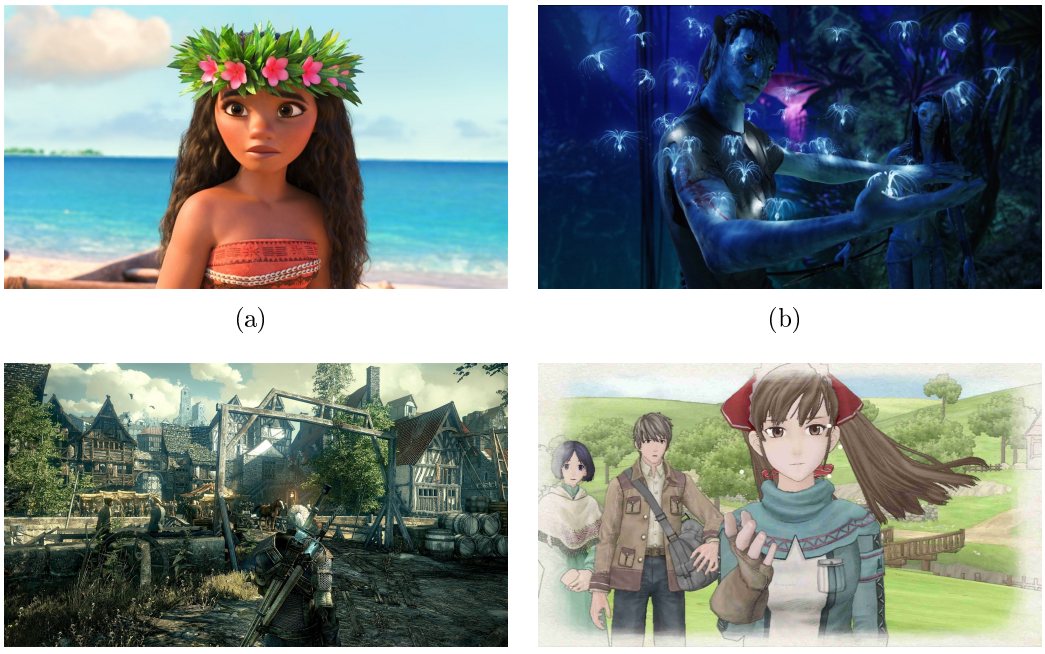
Acronyms

CPU	Central Processing Unit
GPU	Graphics Processing Unit
2D,3D	Two, Three dimensions
CAD	Computer Assisted Design
SDK	Software Development Kit
RTE	Radiative Transfer Equation
NPR	Non-Photorealistic Rendering
BRDF	Bidirectionnal Reflectance Distribution Function
DVR	Direct Volume Rendering
AO	Ambient Occlusion
OIT	Order-Independent Transparency
LHS, RHS	Left, Right Hand Side
FTCS	Forward Time Central Space
FEM	Finite Elements Method
FDM	Finite Differences Method
DOM	Discrete Ordinate Method

Introduction and Motivation

Context and Motivations

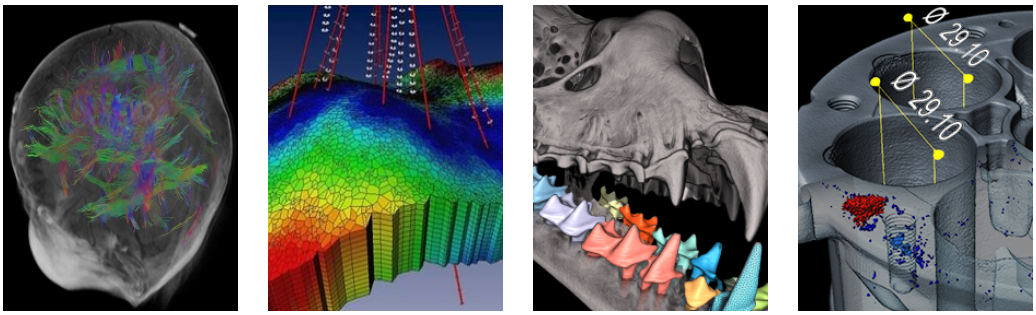
Computer Graphics is the science that includes all the methods to convey visual information using a computer. In particular, image synthesis (or rendering) enables the production of images using a digital scene. This is a widespread process, used in many areas: computer-animated movies (Figure 1(a)), visual effects (Figure 1(b)), video games (Figures 1(c) and 1(d)), scientific visualization...



(a) (b)

(c) (d)

Figure 1 – *Examples of image synthesis in Computer Graphics. (a) A footage of of the computer-animated movie Moana (courtesy of Disney™). (b) A footage of the movie Avatar (courtesy of XXth Century Fox™). (c) Physically-plausible rendering in The Witcher 3 (courtesy of BANDAI NAMCO Entertainment Europe™). (d) Non-photorealistic rendering in Valkyria Chronicles (courtesy of SEGA™).*



(a) (b) (c) (d)

Figure 2 – *Different examples of visualization of scientific data. (a) For medical data: the neurons and synapses in the brain. (b) For seismic analysis: a reservoir. (c) For archaeology: segmenting the tooth of a skull. (d) For industrial inspection: visualizing defaults in a mechanic part. All images are courtesy of Thermo Fisher Scientific™).*

This thesis is conducted within Thermo Fisher Scientific™ to develop demonstrators for the Open Inventor® toolkit. Its main purpose is to provide solutions for the visualization of scientific data. Among the area in which such data are used, we can cite: the medical field (Figure 2(a)), seismic and soil analysis (Figure 2(b)), natural sciences (Figure 2(c)), industrial conception and inspection (Figure 2(d)), etc.

In most of these fields, we often have to rely on the usage of semi-transparent objects (defined either as layers of transparent surfaces or continuous volumes) for exploration and visualization. Addressing transparent objects correctly requires to understand how light is supposed to interact with these objects. It also requires to convey the information in an interactive and legible way. The first issue is addressed by studying light transport theory, the second one by choosing the appropriate rendering method.

The rendering method is also highly dependent on the need of the application and, accordingly, the targeted style of the image is not necessarily the same. Several rendering methods have been designed to address this diversity. For example, physically plausible rendering (Lewis [1993]) consists in depicting as faithfully as possible the effects produced by the light-matter interaction (also called light transport, Veach and Guibas [1997]), as illustrated in Figure 1(c). On the other hand, expressive rendering (Lansdown and Schofield [1995]), also called non-photorealistic rendering (NPR), trades light transport for stylisation, e.g., illustrative style (Gooch and Gooch [2001]), as illustrated in Figure 1(d).

In this thesis, we address both representations, transparent surfaces and, mostly, volumes. We first propose a solution to enhance the legibility of transparent surfaces. We then introduce a new approach using light transport to explore volumes.

Manuscript Organization

This manuscript is divided into four chapters.

Chapter 1 presents the different scientific and technical background that are needed to understand the concepts developed in the manuscript, as well as the problems that we want to address. For this purpose, we discuss the model of light transport applied to volumes and surfaces, and how it is used in Computer Graphics. We then present some techniques of expressive rendering. Finally, we also present Open Inventor, the toolkit from Thermo Fisher Scientific™, in which we have developed the solutions presented in this manuscript.

Chapter 2 targets the specific problems and solutions for rendering transparent surfaces. The main problem is to offer a legible visualization of different layers of transparent surfaces. We introduce our proposition to enhance surface depiction. In particular, we present in this chapter several data structures (including our own) adapted to real-time rendering of transparent surfaces.

Chapter 3 focuses on continuous volumes. As the techniques presented in Chapter 2 are limited when applied to volumes, we introduce a new approach inspired by fluoroscopy imaging in medecine. This new metaphor is based on light transport to enhance depiction in volumes. We also study different numerical approaches in order to determine the characteristics that best suit our application.

Based on these characteristics, Chapter 4 presents the rendering technique we have chosen to solve light transport in volumes for the purpose of expressive exploration. Indeed we introduce how we can use this model to help to identify a region of interest.

All the software contributions and publications are summarized after the conclusion, in a dedicated chapter.

Chapter 1

Rendering Volumes and Surfaces

This chapter is dedicated to the presentation of the prerequisites that are necessary to understand the work presented in this manuscript. This background implies several physical models to describe light transport in different configurations. It also includes the technical details that are needed to implement these models in Computer Graphics.

First, the physical models rely on several radiometric quantities. We then focus on the model that describes the light transport in the case of participating medium, as well as the link between media and surfaces, including the model used for light transport with surfaces. Next, we briefly present how these different models are used in Computer Graphics for image synthesis, with the approximated methods for real-time applications and more physically accurate methods for simulation applications. Finally, we give some details about the Open Inventor SDK, in which the solutions presented in this manuscript have to be developed.

1.1 Prerequisite: Radiometric Quantities

Before presenting any theories and models, we introduce the International System of Units used to measure radiometric quantities. As these radiometric quantities are widely used in this document, knowing them is strongly recommended. Note that in all the quantities presented in this manuscript, all vectors are considered normalized: $||\vec{\omega}|| = 1$, thus we sometimes use the equivalence between the cosinus of an angle $\cos(\theta)$ and the scalar product of the two corresponding vectors ($\vec{u}$, $\vec{v}$):

$$\cos(\theta) = \vec{u}^T \cdot \vec{v}$$

Flux F Also called the radiant power, it represents the total light power, regardless of any spatial or angular distribution. It is expressed in watt **W**, with $1W = 1J.s^{-1}$.

Irradiance E It corresponds to the radiant flux received by a surface, measured per unit area. It is expressed in watt per square meter ($W.m^{-2}$). It relates to the radiant flux with the following equation (where ∂A is a surface unit):

$$E = \frac{\partial F_{incoming}}{\partial A}$$

Radiosity B It corresponds to the radiant flux that leaves a surface, measured per unit area. It is expressed in watt per square meter ($W.m^{-2}$). It relates to the radiant flux with the following equation (where ∂A is a surface unit):

$$B = \frac{\partial F_{outgoing}}{\partial A}$$

Radiance L It corresponds to the radiant flux that crosses a surface for a specific direction, measured per unit solid angle, per unit projected area. It is expressed in watt per steradian (**sr**) per square meter ($W.sr^{-1}.m^{-2}$). It relates to the radiant flux with the following equation (where ∂A is a surface unit and $\partial\omega$ a solid angle unit):

$$L = \frac{\partial^2 F}{\partial A \cdot \partial \omega}$$

1.2 The Radiative Transfer Equation

In this section, we present a theory for light transport in participating media. This theory is based on a statistical approach and leads to a transport equation commonly referred to as the Radiative Transfer Equation (RTE) and was first introduced in the early XX$^{\text{th}}$ century for solving radiative problem (Khvolson [1890] and Schuster [1905]). It was later extended to astrophysics in the 50's (Chandrasekhar [1950]) and neutron transport (Case and Zweifel [1967]). Finally, it was introduced in Computer Graphics by Kajiya and Von Herzen [1984].

To present this transport theory and the different models that are derived from it, we first introduce the phenomena that occur in a participating medium at a mesoscopic scale and then present their physical origin at the scale of a photon. Then, from these phenomena, we establish the Radiative Transfer Equation and finally, we present how it can be reduced to a more practical equation for cases with mostly air and opaque media.

1.2.1 Phenomena in a Participating Medium

We first consider a light beam and focus on its energy. When crossing a participating medium, this beam may be subjected to energy losses or gains.

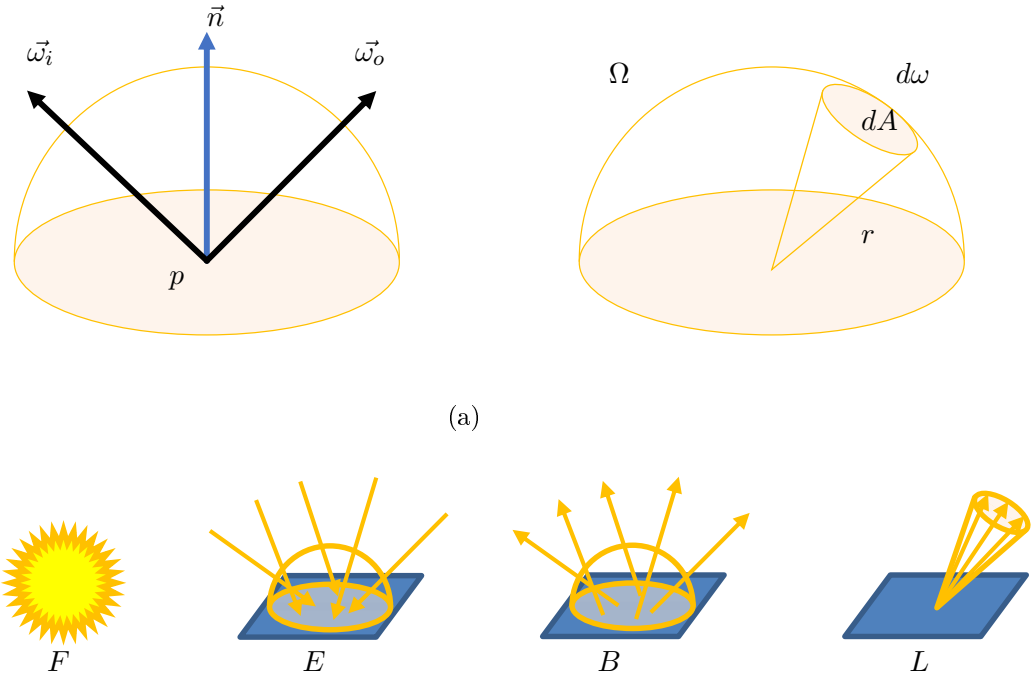
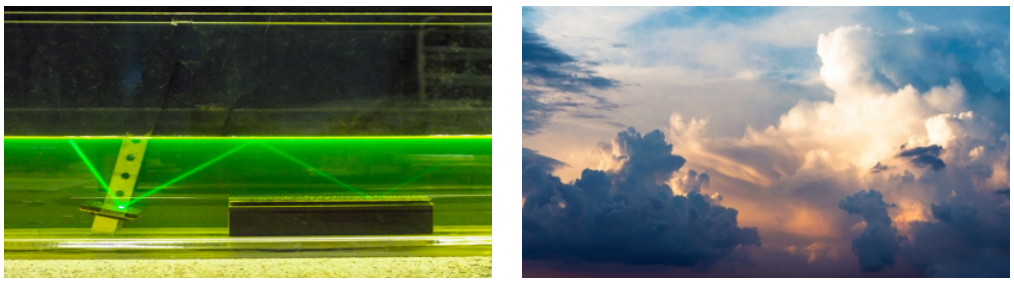


Figure 1.1 – *(a) Left: The notations that are used in this document: $\vec{n}$ is the normal of a surface, $\vec{\omega_i}$ is the incident direction, $\vec{\omega_o}$ is the outgoing direction. Right: the solid angle $d\omega$ in steradian (**sr**) in an hemisphere Ω. (b) The four radiometric quantities used in this document.*



(a) (b)

Figure 1.2 – *Examples of participating media: (a) LASER (535 nm) in a aquarium filled with water mixed with a fluorescein solution, image from education.meteofrance.fr. (b) Sunlight across a cloudy atmosphere.*

These gains and losses are often separated in four phenomena:

1. Emission: energy gain if the media creates some light energy.
2. Absorption: energy loss by energy absorption and conversion.
3. Out-scattering: energy loss by deflection toward another direction.
4. In-scattering: energy gain by deflection from particles nearby.

Examples of real-life participating media in which all these aspects occur are

shown in Figure 1.2. More details on the underlying process of absorption and scattering are presented in Section 1.2.2.

1.2.2 Collision Events and Mean Free Path

Before going into details, please note that the following demonstration is made using a corpuscular approach as it is easier to understand the different phenomena with particles. Note that it can also be established with an ondulatory approach, we refer the reader to Hecht [2002] for an introduction to wave optics.

In the chosen approach, light is composed of energy particles known as photons. The energy that corresponds to a photon is directly linked to its wavelength with the relation:

$$E_{photon} = \frac{h \cdot c}{\lambda}$$

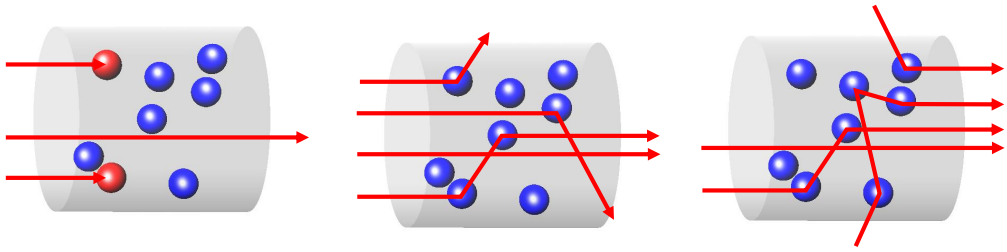
where h is the Planck constant, c is the speed of light in the vacuum and λ is the wavelength. Therefore, whenever we talk about photons, it equivalently refers to: the particle, its energy and its wavelength.

When crossing a medium, a photon may collide with particles. Several phenomena can occur:

1. The photon is absorbed by the particle and converted in another form (heat, electric current...).
2. The photon is absorbed by the particle and re-emitted with the same energy. It may be emitted in any direction, depending on the orientation of the particle in medium. This event can then be interpreted as the photon being deflected.
3. The photon is absorbed by the particle and another one is emitted, with a reduced energy. If the medium is fluorescent, the photon is emitted instantly, whereas if it is phosphorescent, the emission will occur later in time. This event can be separated in absorption (first case) and self-emission.

When dealing with macroscopic information, these events can be modeled as probability densities over a unit distance, related to: the probability of a collision while crossing a certain distance, the probability of this collision resulting in an absorption or the probability of it resulting in a deflection. All events mixed, the probability of a collision is expressed by the Mean Free Path which corresponds to the average distance between two events, in meters.

In the following paragraphs, we present a definition of the two aforementioned cases, i.e., absorption and scattering. The third one can be virtually separated into an absorption event followed by an emission event and thus, it does not need further explanation.



(a) (b) (c)

Figure 1.3 – *(a) Losses due to energy absorption. (b) Losses due to outward deflection (Out-Scattering). (c) Gains due to inward deflection (In-Scattering).*

Absorption This phenomenon is introduced in Figure 1.3(a). Many media, in particular dielectric ones, can absorb photon energy without re-emitting it as light. The absorption coefficient (in m^{-1}), noted K_a, provides a practical way to relate to the probability that a photon is absorbed by a particle when crossing a unit distance. The probability density of a photon being absorbed over a distance l is $e^{-K_a l}$, as stated in the Beer-Lambert law. From this, we can define the absorption length l_{abs}, also known as the absorption Mean Free Path. It corresponds to the characteristic length of an exponential model, thus $l_{abs} = \frac{1}{K_a}$. Note that this length effectively corresponds to the average distance between two absorption events:

$$l_{abs} = \int_0^\infty K_a \cdot s \cdot e^{-K_a s} ds$$

Absorption, if not followed by a re-emission, results in an energy loss for a light beam crossing an absorbing medium.

Scattering The scattering is introduced in Figures 1.3(b) and 1.3(c). Scattering events, when photons are deflected away from their incoming direction, result in an energy loss for a light beam, as illustrated in Figure 1.3(b). But, unlike absorption, scattering events can also cause a light beam to increase in energy. This occurs when neighboring particles deflect photons from other light beams into the one we are considering, as shown in Figure 1.3(c). The first case is often referred to as out-scattering and the second one as in-scattering.

The scattering coefficient is noted K_s (in m^{-1}). As for absorption, we can define the scattering Mean Free Path l_{scat}:

$$l_{scat} = \frac{1}{K_s}$$

When this process occurs for a photon beam, the photons scatter in different directions, resulting in an angular distribution of the scattered energy. This dispersion is quantified by a Phase Function.

Phase Function As stated above, the Phase Function quantifies the portion of the incoming energy that is deflected in a specific direction (van de Hulst [1981]). As it corresponds to an angular distribution, it has a unit of inverse solid angle: sr^{-1}. It depends on the incoming direction (noted $\vec{\omega}_i$), and the outgoing direction (noted $\vec{\omega}_o$). To handle heterogeneity, the phase function also depends on the position (p). In this manuscript, the Phase Function is noted:

$$\boxed{\mathcal{P}(p, \vec{\omega}_i, \vec{\omega}_o)}$$

The phase function must respect several properties to have a physically-compliant behavior:

- **Reciprocity.** It must obey Helmholtz's law of reciprocity, implying that:
$$\mathcal{P}(p, \vec{\omega}_i, \vec{\omega}_o) = \mathcal{P}(p, \vec{\omega}_o, \vec{\omega}_i)$$
- **Energy conservation.** Or normalization, meaning that no energy can be created in the scattering process, thus the phase function must integrate to one:
$$\int_{\Omega^2} \mathcal{P}(x, \vec{\omega}_i, \vec{\omega}_o) d\vec{\omega}_o = 1$$
Note that depending on the papers and the convention chosen, the phase function sometime integrates to 4π.

The second rule gives the expression of the uniform phase function:

$$\mathcal{P}(p, \vec{\omega}_i, \vec{\omega}_o) = \frac{1}{4\pi}$$

For non-uniform scattering, the Henyey-Greenstein phase function (Equation 1.2) is used as it can approximate most scattering behaviors (forward and backward scattering). This is done by using the Anisotropy Factor g (Equation 1.1) varying from -1 to 1: $g < 0$ corresponds to backward scattering, $g = 0$ is the uniform phase function and $g > 0$ corresponds to forward scattering.

$$g(p) = \frac{1}{4\pi} \int_{\Omega^2} \mathcal{P}(p, \vec{\omega}, \vec{\omega}') \vec{\omega}^T \cdot \vec{\omega}' d\omega' \tag{1.1}$$

$$\mathcal{P}(p, \vec{\omega}_i, \vec{\omega}_o) = \frac{1 - g^2}{\sqrt{1 + g^2 - 2g\vec{\omega_i}^T \cdot \vec{\omega_o}}^3} \tag{1.2}$$

However, for really accurate physical simulation, one should use the phase function that is suited to the physical configuration (e.g., the Rayleigh phase function for scatterers smaller than the wavelength, see van de Hulst [1981] or Hecht [2002] for more details). Also, note that in this manuscript, the effects of correlation in the medium particle distribution is hidden in the phase function, and we do not detail its impact on the scattering process. For further details, we refer the reader to Jarabo *et al.* [2018], who present the impact of different levels of correlation for a medium composed of the same type of particles.

Emission

The emission inside a medium can be viewed as the inverse of absorption: instead of light energy being absorbed and converted into heat, a heated medium decreases its energy by converting it into photons.

This process is often approximated using the black-body radiative model, first introduced by Kirchhoff and Bunsen [1860]. This model introduces a relation between the temperature of a medium (in Kelvin **K**) and the wavelength of the photons that are emitted in order to decrease its energy. The relation between the temperature T and the peak wavelength λ_{max} is given by the Wien's Law (Wien [1894]):

$$\lambda_{max} = \frac{b}{T}$$

with b being the Wien's displacement constant, equal to approximately $2.897 \times 10^{-3} m.K$.

If Wien's Law gives the peak wavelength, Planck and Masius [1914] introduce a relation that gives the radiance emitted by a medium at temperature T for a specific wavelength λ:

$$L(\lambda, T) = \frac{2h}{\lambda^3 c^2} \frac{1}{e^{\frac{h}{\lambda k T}} - 1}$$

where $c \approx 3.10^8 m.s^{-1}$ is the speed of light in vacuum, $h \approx 6.626 \times 10^{-34} J.s$ is the Planck constant and $k \approx 1.381 \times 10^{-23} J.K^{-1}$ is the Boltzmann constant.

For the remainder of this document, we use a notation more suited to Computer Graphics applications. The contribution of emissive media is described by a volumetric term Q_e, expressed in watt per cube meter per steradian ($W.m^{-3}.sr^{-1}$).

1.2.3 Establishing the Radiative Transfer Equation

Now that the phenomena occurring in participating medium have been introduced, we can focus on the formulation of the Radiative Transfer Equation, as introduced in Chandrasekhar [1950] and its formulation as used in Computer Graphics.

The Differential Form

To estimate the influence of these phenomena, we first focus on an elementary part of a volume, of length $\delta s = c \cdot dt$, where dt is a time step, aligned with the direction $\vec{\omega}$, as illustrated in Figure 1.4.

Before entering the subvolume, at a point p, we have a radiance $L(p, t, \vec{\omega})$ and an exiting radiance of $L(p + \vec{\omega}\delta s, t + dt, \vec{\omega})$. Inside the sub-volume, a portion of $L(p, t, \vec{\omega})$ is absorbed, another scattered. At the same time, the

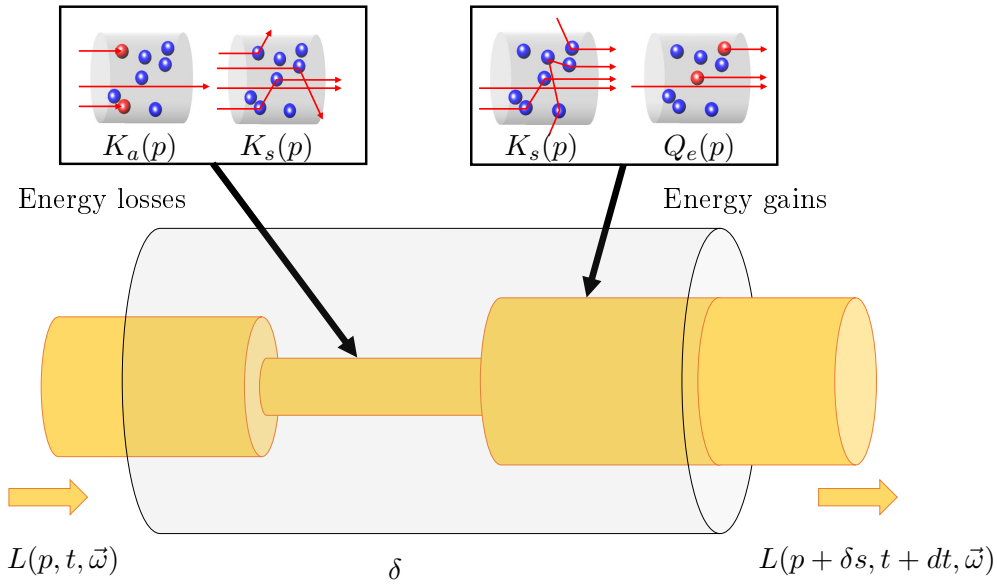


Figure 1.4 – *Propagation through a sub-volume $\delta s = c{\cdot}dt$. The incoming light beam of radiance $L(p,t,\vec{\omega})$ looses energy due to out-scattering (K_s) and absorption (K_a), and in the meantime, it gains energy by in-scattering (K_s) and potential emission (Q_e).*

sub-volume may emit energy, and the surrounding particles also scatter energy that contributes to the sub-volume exiting radiance. The contributions along δs are then:

- $-c \cdot dt \cdot K_a(p) \cdot L(p,t,\vec{\omega})$, the absorption along δs.
- $-c \cdot dt \cdot K_s(p) \cdot L(p,t,\vec{\omega})$, the out-scattering along δs.
- $+c \cdot dt \cdot Q_e(p,t,\vec{\omega})$, the volumetric emission along δs (in $W.m^{-3}.sr^{-1}$).
- $+c \cdot dt \cdot K_s(p) \int_{4\pi} \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,t,\vec{\omega'}) d\vec{\omega}'$, the incoming energy from neighborhood along δs (aka in-scattering).

As absorption and out-scattering both imply a loss of energy, they are regrouped under an extinction term. The associated extinction coefficient noted K_t is defined as:

$$K_t = K_a + K_s$$

We can obtain the exiting radiance with an energy balance between p and $p + \delta s$:

$$\begin{aligned} L(p+\vec{\omega}\delta s, t+dt, \vec{\omega}) &= L(p,t,\vec{\omega}) - c \cdot dt \cdot K_t(p) \cdot L(p,t,\vec{\omega}) \\ +c \cdot dt \cdot Q_e(p,t,\vec{\omega}) &+ c \cdot dt \cdot K_s(p) \cdot \int_{4\pi} \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,t,\vec{\omega}') d\vec{\omega}' \end{aligned} \tag{1.3}$$

By using the Euler-Lagrange relation (in our case $\frac{d}{dt} = \frac{\partial}{\partial t} + c\vec{\omega}^T \cdot \vec{\nabla}_p$, as stated in Pierrat [2007]), this equation can be written as a differential equation,

also known as the Radiative Transfer Equation (RTE).

$$\boxed{\begin{aligned}\frac{1}{c}\frac{\partial L(p,t,\vec{\omega})}{\partial t} + \vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega}) &= -K_t(p) \cdot L(p,t,\vec{\omega}) + Q_e(p,t,\vec{\omega}) \\ &+ \int_{4\pi} K_s(p) \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,t,\vec{\omega}')d\vec{\omega}'\end{aligned}} \tag{1.4}$$

For a more physically accurate and more detailed demonstration about this equation, the reader is encouraged to read Pierrat [2007], who establishes this equation and its implications, as well as a demonstration from Maxwell laws on electromagnetic wave propagation.

Equation 1.4 is the general definition of the RTE. However, in Computer Graphics, we are often in a situation where we can consider to be in a local thermodynamic equilibrium, meaning that the time-related dependence becomes irrelevant. Thus, if not otherwise specified, the steady version (Equation 1.5) is the one we refer to for the rest of this chapter.

$$\boxed{\begin{aligned}\vec{\omega}^T \cdot \vec{\nabla}_p L(p,\vec{\omega}) &= -K_t(p) \cdot L(p,\vec{\omega}) + Q_e(p,\vec{\omega}) \\ &+ \int_{4\pi} K_s(p) \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,\vec{\omega}')d\vec{\omega}'\end{aligned}} \tag{1.5}$$

The Integral Form

We established the differential form of the RTE. In some cases, a formulation that accounts for a whole light ray is more practical. Equation 1.5, on a sub-volume, has the form:

$$y'(p) + A(p) \cdot y(p) = q(p)$$

where:

$$A(p) = K_t(p)$$

$$q(p) = Q_e(p,\vec{\omega}) + \int_{\omega} K_s(p) \cdot \mathcal{P}(p,\omega,\omega') \cdot L_i(p,\vec{\omega}')d\omega'$$

The solution of this differential equation is of the form:

$$y(p) = C \cdot e^{-\int^p A(s)\cdot ds} + \int^p q(v) \cdot e^{-\int_v^p A(u)\cdot du} \cdot dv$$

In our case, the solution for a ray going from a point p_0 to p is then:

$$L(p,\vec{\omega}) = L_0(\vec{\omega}) \cdot e^{-\int_0^p K_t(u)\cdot du}$$
$$+ \int_0^p \left(Q_e(t,\vec{\omega}) + \int_{4\pi} K_s(v) \cdot p(v,\vec{\omega},\vec{\omega}') \cdot L_i(v,\vec{\omega}')d\vec{\omega} \right) \cdot e^{-\int_v^p K_t(u)\cdot du} dv$$

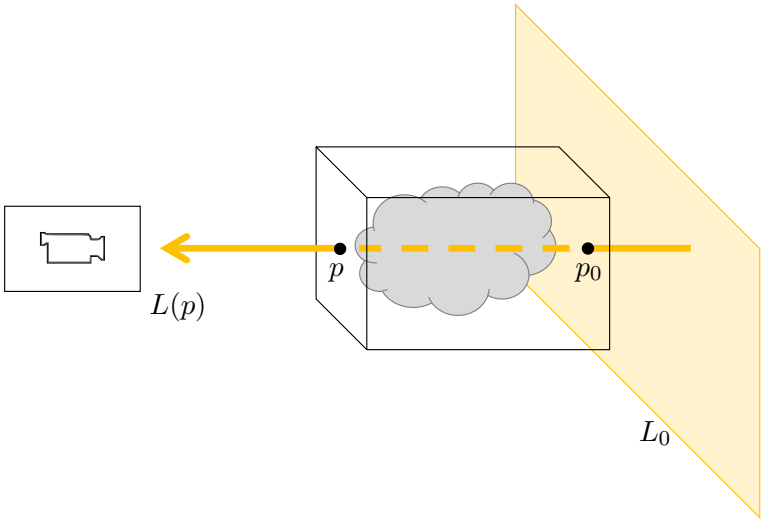


Figure 1.5 – *A cloud with its bounding box in front a light panel. Light emitted from the panel, with radiance L_0, enters the volume at position p_0 with radiance $L(p_0) = L_0$ and propagates through it, emerging at position p. The radiance at position p is given by Equation 1.8.*

Emission and in-scattering can be regrouped under an "energy gain" function $Q(p, \vec{\omega})$ such as:

$$\boxed{Q(p,\vec{\omega}) = Q_e(p,\vec{\omega}) + K_s(p)\int_{4\pi} \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,\vec{\omega}')d\vec{\omega}'} \tag{1.6}$$

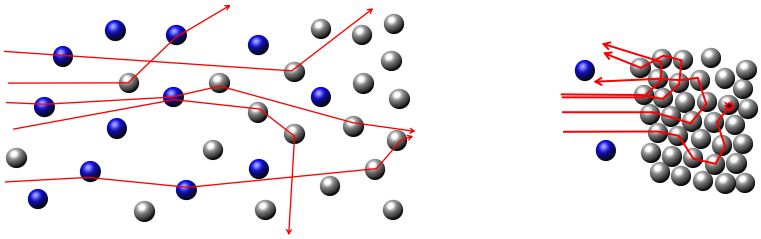
For convenience, we use $\tau(t_1, t_2)$, the cumulative absorption from t_1 to t_2, also known as the transmission rate of the slab (t_1, t_2).

$$\tau(t_1,t_2) = e^{-\int_{t_1}^{t_2} K_t(s)\cdot ds} \tag{1.7}$$

Finally, for a full volume, placed in front of a uniformly lit background (position p_0), as illustrated in Figure 1.5, the radiance received at each point on the plane at position p obeys the following Equation (1.8).

$$\boxed{L(p,\vec{\omega}) = L_0 \cdot \tau(p_0,p) + \int_{p_0}^{p} Q(u)\cdot\tau(u,p)\cdot du} \tag{1.8}$$

This integral form (Equation 1.8) is sometimes referred to as the Volume Rendering Equation in the Computer Graphics community.



(a) (b)

Figure 1.6 – *(a) Smooth transition between two media resulting mostly in a forward diffusive pattern. (b) Discontinuous transition between two media with significant disparities, resulting in a reflective pattern. Note that depending on the type of medium and its structure (e.g., solid state), it could result in a refractive pattern (like glass).*

1.2.4 Interfaces between Media: toward Surfaces

We have explained and detailed the model for light transport in a participating medium. We now focus on what happens when we consider a configuration with different media. Such a configuration could for example originate from having a variation of density, a variation of the medium composition (like air and water vapor), etc. We can separate this problem in two categories: smooth transition between the media or discontinuous transition. To distinguish between the two, we can look at the variation of the RTE parameters in regard to their scale, that is to say, compare the order of $\nabla K_{a/s/t}$ with the order of $K_{a/s/t}$.

Smooth Transitions This case corresponds to situations where the variations of the parameters are very small in regards to the order the parameters:

$$\frac{\nabla K_{a/s/t}}{K_{a/s/t}} \ll 1$$

Thus, the parameters can be considered locally uniform and then, the problem can be reduced to solving the RTE with a heterogeneous medium. This case is illustrated in Figure 1.6(a).

Discontinuous Transitions: Interfaces In any other cases with sharp transition, we need to introduce interfaces to model the transition points. If the transition is nearly perfectly sharp (like air and glass), we only need one interface to describe the changes. This case is illustrated in Figure 1.6(b).

This notion will prove to be useful when dealing with configuration combining air with extremely dense medium (like a metal). Indeed, for configurations like a room or a street, most of the space is composed of air (very low-absorption and very low-dispersion at the considered scale), while the rest is most likely composed of dense media like painted concrete, plastic, glass,

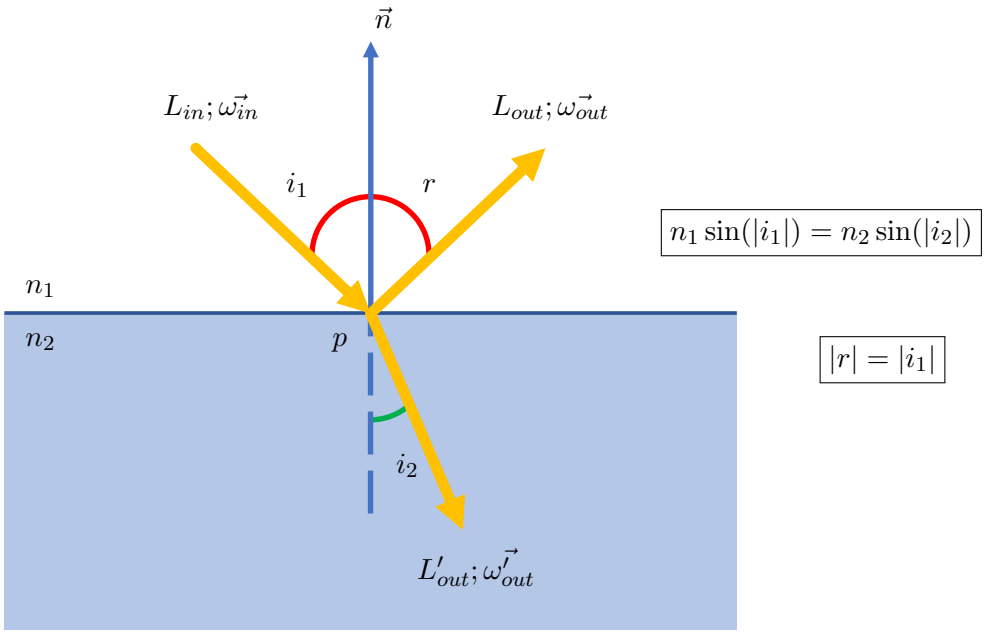


Figure 1.7 – *An interface between a medium of index n_1 and one of index n_2 with reflection and refraction. A portion of the incoming light ($L_{in};\vec{\omega_{in}}$) is reflected $L_{out};\vec{\omega_{out}}$ according to the Fresnel Coefficient R such as $L_{out} = RL_{in}$, while the other portion is refracted ($L'_{out};\vec{\omega'_{out}}$) according to the Fresnel Coefficient T such as $L'_{out} = TL_{in}$. Note that $R+T = 1$. The reflection and refraction angles are obtained with the Snell-Descartes law, on the right of the figure.*

metals, etc. The air can then be approximate as vacuum, with nearly no impact on light transport. Another approximation that can be made here is to consider that, due to the nature of the other media involved, light is either completely reflected or absorbed by the media (metal, plastics), or crosses it without loss or dispersion (glass).

By using these approximations, this type of scene can be reduced to the different medium interfaces, and thus by surfaces corresponding to these interfaces. In this case, the RTE is no longer used as its complexity is not suited to the resolution of a problem that becomes mostly addressed by Geometric Optics (treating light as rays, obeying the Fresnel laws for reflection and transmission coefficient and Snell-Descartes for reflection and refraction angles, see Figure 1.7).

The Rendering Equation

When dealing with a situation modeled by surfaces, with ray optics, Kajiya and Von Herzen [1984] introduced the Rendering Equation:

$$\boxed{L(p, \vec{\omega}_{out}) = L_0(p) + \int_{pi} f_r(p, \vec{\omega}_{out}, \vec{\omega}_{in}) \cdot L_i(p, \vec{\omega}_{in}) \vec{\omega}_{in}^T \cdot \vec{n} d\vec{\omega}_{in}} \tag{1.9}$$

In Equation 1.9, the function f_r quantifies the portion of energy that is reflected (or refracted) from direction $\vec{\omega}_{in}$ toward direction $\vec{\omega}_{out}$ at position p, and $\vec{n}$ is the normal vector of the surface at position p. Even if this equation still has a recursive formulation, it is much lighter to evaluate since it only requires to be evaluated for light sources and their reflections.

Note the scalar product term $\vec{\omega}_{in}^T \cdot \vec{n}$ in the integral. As an incoming light beam may not be perpendicular to the interface, its energy is evenly distributed on an area larger than the cross-section of the beam.

Bidirectional Reflectance Distribution Function

As stated above, in Equation 1.9, function f_r quantifies the ratio of energy that a surface reflects in a given direction depending on the energy it receives from another direction. It is a 4D function called the Bidirectional Reflectance Distribution Function (BRDF) and was introduced by Nicodemus [1970] and is defined as:

$$\boxed{f_r(\vec{\omega}_{out}, \vec{\omega}_{in}) = \frac{dL_r(\vec{\omega}_{out})}{dE_{in}(\vec{\omega}_{in})} = \frac{dL_r(\vec{\omega}_{out})}{L_i(\vec{\omega}_{in})(\vec{\omega}_{in}^T \cdot \vec{n}) d\vec{\omega}_{in}}} \tag{1.10}$$

Like the phase function, the BRDF, to be physically-compliant, must respect several conditions:

- **Positivity.** Radiance is positive, thus:

$$f_r(\vec{\omega}_{in}, \vec{\omega}_{out}) \geq 0$$

- **Reciprocity.** It must obey Helmholtz's law of reciprocity:

$$f_r(\vec{\omega}_{in}, \vec{\omega}_{out}) = f_r(\vec{\omega}_{out}, \vec{\omega}_{in})$$

- **Energy conservation.** Or normalization, meaning that no energy can be created by the reflection process:

$$\int_{\pi} f_r(\vec{\omega}_{in}, \vec{\omega}_{out}) d\vec{\omega}_{out} \leq 1$$

It is important to note that, by definition, any phenomenon mentioned in Section 1.2.1 that occurs behind the surface (like scattering or absorption) is

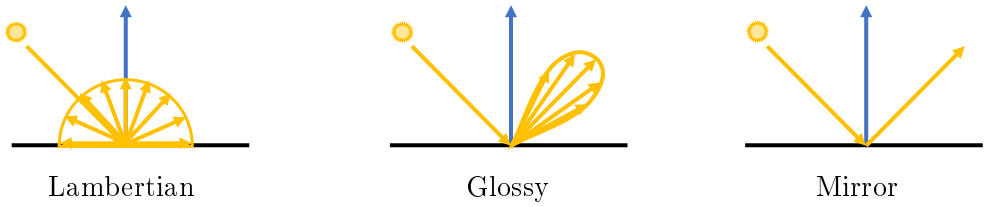


Figure 1.8 – *The three BRDF behaviors: pure Lambertian, glossy and pure specular (mirror).*

embedded directly in the BRDF. Thus, three categories of reflections (as illustrated in Figure 1.8) can be modeled with a BRDF: the pure specular one (mirror-like, case of the Figure 1.7), the pure Lambertian one (uniform reflection) and the glossy reflection with a lobe distribution. The BRDF is often simplified as a diffuse Lambertian term and a specular one (e.g., Cook and Torrance [1982], Ashikhmin and Shirley [2000], Burley and Walt Disney Animation [2012]). The first one accounts for reflected energy coming from internal scattering behind the surface, while the second one accounts for the energy reflected directly by the surface. As for the phase function, there would be much to say about BRDF, which falls out of the scope of this thesis. The interested reader can see the overview from Montes and Ureña [2012].

1.3 Rendering Techniques

The simulation of a lighting model for visualizing three-dimensional scenes is an essential tool for understanding the content of this scene. In particular, reflections are crucial in understanding a shape Fleming *et al.* [2004]. Image synthesis in Computer Graphics consists in simulating this light transport to convey these features.

The easiest and most affordable way to do this is to mimic Geometric Optics: shooting and tracing a ray from a virtual camera across the scene and computing the bounces on the different interfaces as well as the effect of scattering in case of participating media. This approach, called ray-tracing (Whitted [1980]), is one of the oldest in Computer Graphics, however, it implies a huge computation cost. Thus, other paradigms have been developed, in particular with the development of dedicated Graphical Processing Units (GPU). Their aim is to provide faster but less physically-accurate rendering for applications that require interactivity more than accuracy. Even if many techniques have been proposed to enhance the quality of these rendering methods, ray-tracing remains the best paradigm for high quality images with global illumination.

In this section, we present these aspects: first an overview of the graphic

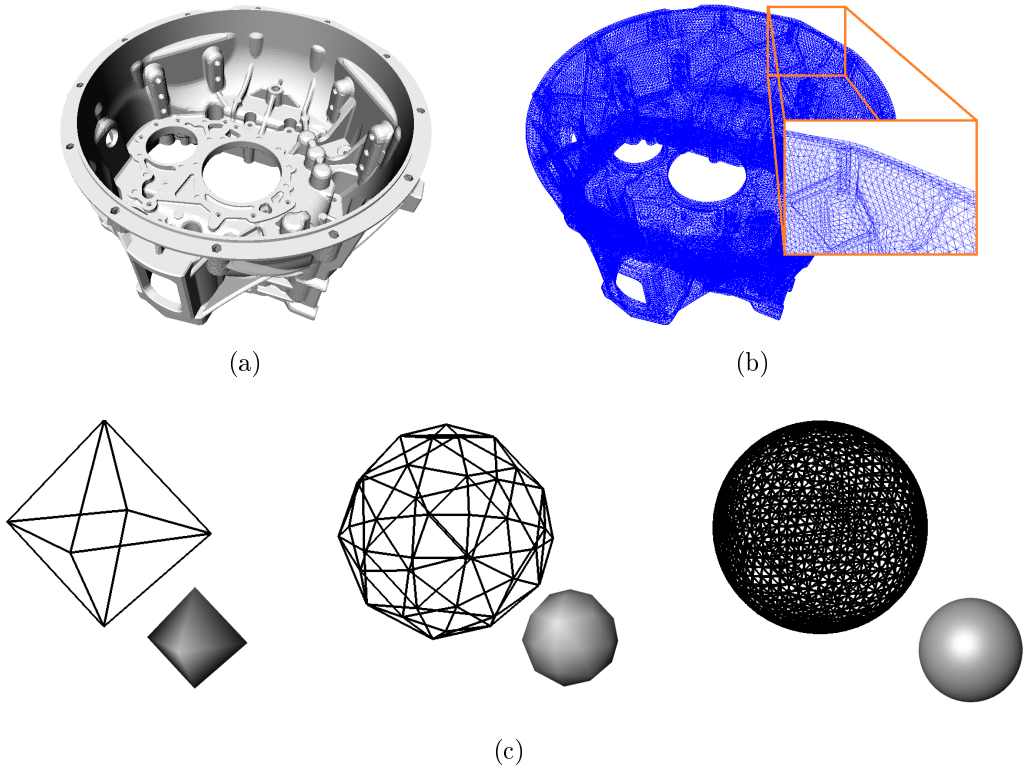


Figure 1.9 – *An object (a) and its representation as a triangular mesh (b). (c) Several possible mesh representations of sphere, depending on the necessary refinement: symbolic sphere (left), trade-off between representation and memory occupancy (center), accurately represented sphere (right).*

pipeline (used for surfaces) and the pipeline used for volumes, then several simple lighting solutions, to complement the previous techniques, and finally techniques that embed the lighting computation into the rendering paradigm.

1.3.1 Rendering for Surfaces

Surface Representation

In Computer Graphics, surfaces are generally discretized as polygons, represented by their vertices, as illustrated in Figure 1.9. In practice, even though one could use any polygonal representation, the preferred one is triangles as a dedicated pipeline (presented in Section 1.3.1) was developed on GPU for triangular meshes.

This representation offers a good trade-off between memory usage (the number of required elements) and the level of details that can be represented. Indeed, a sphere can be represented with 8 triangles if its purpose is purely illustrative and does not require an accurately rounded shape. It can also be much more refined with hundred of triangles if its purpose is to offer a more

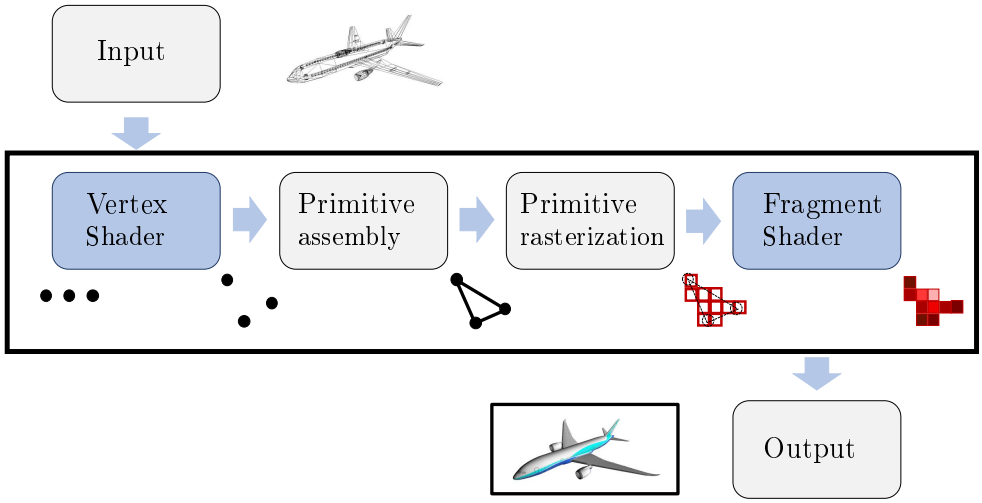


Figure 1.10 – *Graphics pipeline used in classical surface-based GPU rendering.*

accurate shape, as illustrated in Figure 1.9(c).

Graphics Pipeline

Rasterization is a process that produces a 2D matrix image from a 3D representation (or a vector-defined image) by projecting the latter on a 2D plane. This projection can be done with a CPU. However, as this process scales well with highly parallel architectures, like a GPU, it has quickly become a standard for real-time rendering algorithms on GPU, while also contributing to the promotion of the GPU as a key hardware in Computer Graphics.

Indeed, a special pipeline was developed, optimized and even integrated in the development of GPUs, which is often referred to as the graphics pipeline, presented in Figure 1.10. This process also has evolved along with the refinement of GPUs, that offer more computing power each year, and now consists in sending the input data on the GPU, applying transformation on the 3D data, projecting it onto the 2D plane, computing each pixel's color, returning the final image.

Note that in Figure 1.10, some steps are pictured in blue boxes. These steps are programmable using Shaders, small programs executed directly on the GPU. There is one type of shader for each element. Vertex shaders handle the transformation that must applied to each vertex to correctly place them in the scene. These points are then rasterized into fragments (non programmable step) which are handled by fragment shaders to determine the color of the final pixel. The development of shaders is another reason for the success and

the spreading of the graphics pipeline. Note that other shader stages exist for geometry processing (geometry shaders and tesselation shaders) and an external one for computing purpose (compute shaders). For a more complete overview of this whole process and the different shader stages, we refer the reader to Akenine-Möller *et al.* [2018].

Finally, to implement post-processing algorithms, it is possible to store the result of the graphics pipeline (called a Frame Buffer Object). The desired post-processing effect can then be applied on top of the rendering obtained previously. Using this two-pass approach is called deferred rendering, as opposed to forward rendering (one pass using the graphics pipeline).

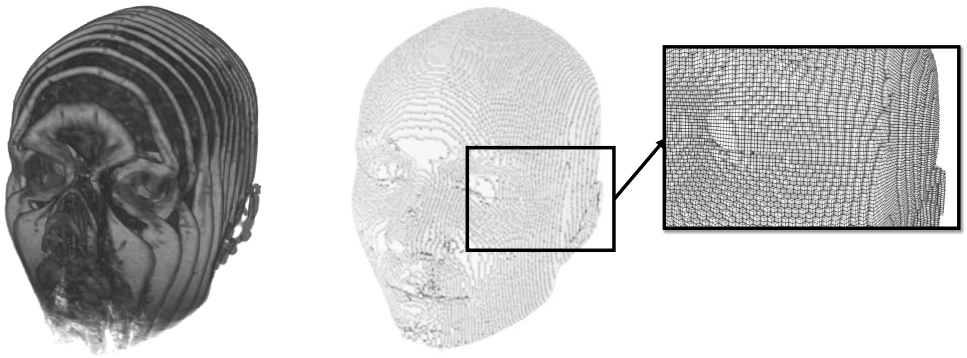
1.3.2 Volume Rendering

Volume rendering is typically done by solving the Volume Rendering Integral, presented in Equation 1.8. This requires to discretize the integral and evaluate it for a set of sampled positions. But before presenting how it is solved in Computer Graphics, we first present the representation that is used to handle the data. Then we present the pipeline used to compute the volume rendering integral, as well as the mostly used method for volume visualization. Then, we present how the Volume Rendering Integral is discretized to comply with this method. Finally, we briefly present the concept of isosurface, a special case in handling volumes.

Volume Data

To understand how volumes are represented in Computer Graphics, we must first focus on the origin of these data. There are several fields in which volume data are used: biomedical, seismic analysis, life science, industrial inspection. Some of them use the process of Computed Tomography (CT) scanning with X-rays. This process consists in doing multiple X-ray measurements from different angles, thus producing cross-sectional images (also referred to as "slices") of specific areas. Slices allow the user to inspect parts inside the object (organs, electronic circuits...) in a non-invasive way. Slices can also be obtained from Magnetic Resonance Imaging (MRI) when inspecting for brain injuries. Note that the content of the slices corresponds to a physical quantity depending on the acquisition device: transmittance, reflectance, etc. In most cases, the information is limited to a scalar value per position.

To explore the acquired data, in particular for medical imaging, one can either use directly the slices and observe them one by one, or generate an image from a 3D reconstruction of the slices. Indeed the set of slices can be reassembled to form a 3D matrix (a volume), as illustrated in Figure 1.11. The volume elements of the resulting matrix are called "voxels" (3D pixels). Acquisition and reconstruction are entire fields of study that are out of the



(a) (b)

Figure 1.11 – *The same volume represented as: (a) a set of slices. (b) voxels.*

scope of this manuscript. For an introduction on these processes, we refer the reader to Hadwiger *et al.* [2006].

Volume Rendering Pipeline

If observing any slice directly from the acquired data goes back to visualizing an image, the problem is different when we need to explore the whole data. If we are dealing with a set of slices, we need to compose them with each other, and if we are dealing with a reconstructed volume, we need to compose its voxels. As both interpretations rely on a compositing scheme, the same pipeline can be used for both a set of slices and a volume. This pipeline (presented in Hadwiger *et al.* [2005]) is composed of the following steps:

Data traversal Sampling positions are chosen to serve as a discretization pattern to compute the volume rendering integral. Data information is then queried at these positions, and interpolated if necessary.

Gradient computation Some shading algorithms and lighting simulations may require the gradient of the scalar field. It is usually computed using central finite differences.

Classification The data queried from the volume are scalar values that originated from a physical measure, thus it must be mapped to a color and opacity value. This step is crucial as it allow us to distinguish between the different areas of the volume. The mapping is usually done using a Transfer Function (Ljung *et al.* [2016]).

Shading and illumination Shading and lighting can be incorporated in the computation of the volume rendering.

Compositing This step controls the iterative computation of the volume-rendering integral. Once the samples have been assigned a color and opacity, they are composed with each other. The compositing pattern is described in Section 1.3.2.

Regarding this pipeline, the main difference between the two data representations lies in the Data Traversal step. Thus, several volume rendering techniques were proposed, using either slices or volumes. However, one has been predominant for the last decade: the direct volume rendering, which is the subject of the next paragraph.

Direct Volume Rendering: Ray-casting

Several techniques have been used to render volumes. In this thesis, we focus on the one called Direct Volume Rendering. For an overview of the other existing techniques, we once again refer the reader to Hadwiger *et al.* [2006].

Direct volume rendering relies on a method called ray-casting. Ray-casting is a technique that consists in evaluating the volume rendering equation (Equation 1.8) with a front-to-back ray-marching process: for each pixel in the image plane, a ray is cast toward the volume and samples are accumulated and blended with a discrete marching step, as presented in Figure 1.12. The blending operator can be obtained by discretizing the volume rendering equation, as shown in the next paragraph. Ray-casting was first used for CPU volume rendering, but the introduction of shaders on GPU allowed efficient implementation of ray-casting on GPU and real-time volume rendering.

The Discrete Volume Rendering Equation

Several models of $Q(p, \vec{\omega})$ (Equation 1.6) can be used in volume rendering but the vast majority relies on a emission-absorption model: $K_t = K_a$, and $Q(p, \vec{\omega}) = Q_e(p, \vec{\omega})$. This simplification comes from the heavy computational cost of the scattering part, which implies a recursive spherical integration of all surrounding contributions. Thus, with this model, Equation 1.8 can be rewritten:

$$L(p, \vec{\omega}) = L_0 \cdot \tau(p_0, p) + \int_{p_0}^{p} Q_e(u) \cdot \tau(u, p) \cdot du$$

Until now, we have considered light traveling from the light source toward the sensor (camera, eye...), as illustrated in Figure 1.5. However, doing the composition from the sensor is more efficient as only elements actually contributing to the final image are taken into account. Thus, we need to compute the inverse path: a light is cast from the sensor toward the volume, as illustrated in Figure 1.12. That is to say, we compute $L(p, \vec{\omega})$ by going from p to p_0, in the direction $-\omega$ instead of ω. As, for a defined finite interval, $|\int_y^x| = |\int_x^y|$,

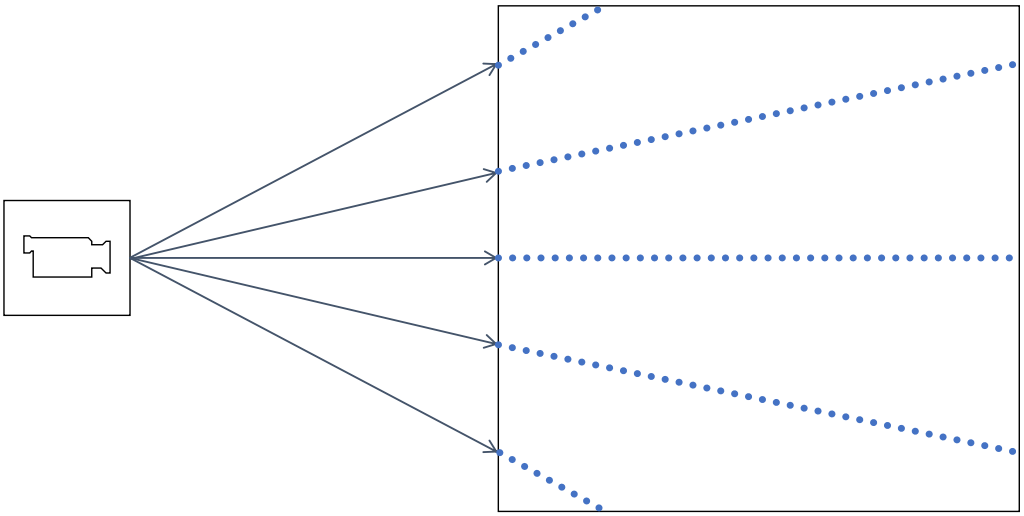


Figure 1.12 – *Ray-marching process to sample a volume. At each sample, the data is queried to evaluate the volume rendering equation (using gradients, classification and shading). The result is then blended with previous samples with front-to-back compositing.*

the resulting equation is then:

$$\boxed{L(p,\vec{\omega}) = L_0 \cdot \tau(p,p_0) + \int_p^{p_0} Q_e(u) \cdot \tau(p,u) \cdot du} \tag{1.11}$$

This equation can be discretized to obtain a blending pattern for compositing, presented in Equation 1.13 for computing colors and Equation 1.14 for transparency. This is obtained using $L(p) \Longleftrightarrow L_i$, with $i = 0$ corresponding to the entry point into the volume (at position p), and $i = n$ corresponding to the exit point (p_0). Thus:

$$\int_p^{p_0} dx \Longleftrightarrow \sum_{i=0}^{n} \Delta x$$

Furthermore, for simplification, we omit here the term coming from a potential backlight, which leads to:

$$L(p,\vec{\omega}) \Longleftrightarrow L_n = \sum_{i=0}^{n} Q_{e,i}\Delta x \cdot \tau_i$$

with:

$$\tau_i = e^{-\sum_{j=0}^{i} K_{a,j}\Delta x} = \prod_{j=0}^{i} e^{-K_{a,j}\Delta x}$$

This formula can be simplified if we note that $e^{-K_{a,i}\Delta x}$ corresponds to the transmittance of a voxel of size Δx. Note that $e^{-K_{a,i}\Delta x} = 0$ means that the voxel does not transmit any light, it is opaque, whereas $e^{-K_{a,i}\Delta x} = 1$ means that it transmits all the light it receives, it is then transparent. The transmittance can be interpreted as the "transparency" of the voxel. The relation to "opacity", a notion more frequently used in Computer Graphics (noted α) is: $\alpha = 1 - e^{-K_{a,i}\Delta x}$. We now use this notation and we have:

$$\tau_i = \prod_{j=0}^{i}(1 - \alpha_i)$$

The final composition is described by Equation 1.12.

$$\boxed{L_n = L_0 \cdot \tau_n + \sum_{i=1}^{n} Q_{e,i}\Delta x \tau_i} \tag{1.12}$$

To obtain the elementary version, which can be used for the ray-casting process, we need to decompose this equation:

$$L_n = \sum_{i=0}^{n-1} Q_{e,i}\Delta x \cdot \prod_{j=0}^{i}(1 - \alpha_j) + Q_{e,n}\Delta x \cdot \tau_i$$

$$L_n = L_{n-1} + Q_{e,n}\Delta x \cdot \tau_n$$

If we note:

$$C_{out} = L_n; C_{in} = L_{n-1}; C = Q_{e,n}\Delta x$$

$$\alpha = \alpha_n; \alpha_{in} = 1 - \tau_{n-1}; \alpha_{out} = 1 - \tau_n$$

We then have:

$$\boxed{C_{out} = C_{in} + (1 - \alpha_{in})C} \tag{1.13}$$

Using the same decomposition scheme as for L_n, we can write the complementary version for α_{out}:

$$\boxed{\alpha_{out} = \alpha_{in} + (1 - \alpha_{in})\alpha} \tag{1.14}$$

Equations 1.13 and 1.14 correspond to the UNDER blending operator. This operation is the front-to-back equivalent of the OVER operator, the back-to-front blending operator described by Porter and Duff [1984]. Note that this result could also be extracted directly from Equation 1.4 with the Emission-Absorption model.

A Special Case: Isosurfaces When dealing with volume data, one may want to see only its significant interfaces. These interfaces result in isosurfaces as they correspond to voxels with the same scalar value. As just identifying these surfaces using scalar values is uncertain due mostly to the need of a tolerance threshold, and the fact that the data is reconstructed (possibly noisy data), some techniques have been proposed to enhance this process (e.g., Tatarchuk *et al.* [2007]). In some cases, the isosurfaces can be extracted and converted in a surface mesh, and later used with the graphics pipeline in another application.

1.3.3 Illumination techniques

This section focuses on the techniques used to evaluate illumination in a scene, using either the graphics pipeline or the volume rendering pipeline. Due to hardware limitations, in particular for interactive applications, illumination simulation was first limited to direct lighting (only one interaction between the light and the scene). The result of multiple reflections or scattering events, called indirect lighting, was either embedded into the model or precomputed. However, light transport implies potential occluders on the light path. If this case is straightforward with ray-tracing techniques, it requires additional considerations for the other techniques.

Direct Lighting

We now focus once again on our two rendering equations: Equation 1.8 for volumes and Equation 1.9 for surfaces. We detail how these two equations are solved when we consider only the direct contribution of discrete light sources.

To account for direct lighting only (no bounces), the integral in Equation 1.9 can be discretized according to the number of light sources. Note that at this point we only considered direct lighting from a discrete set of light sources. In this case, indirect lighting must be approximated. The easiest way of doing so is to add an ambient contribution, $L_a(p)$, in the direct lighting computation such as:

$$L(p, \vec{\omega}_{out}) = L_e(p) + L_a(p) + \sum_{i=0}^{\#lights} f_r(p, \vec{\omega}_{out}, \vec{\omega}_i) \cdot L_i(p, \vec{\omega}_i) \vec{\omega}_{in}^T \cdot \vec{n}$$

The same can be done for Equation 1.8 by adding some simplifications. As stated in Section 1.3.2, we use the Emission-Absorption model. However, light contributions are added to the emission term. In order to keep a low computation cost, no loss or scattering are taken into account between the light and the point. Thus, to compute these contributions, we must determine if the light should be reflected or scattered, as stated in Hadwiger *et al.* [2009]. To do so, we can consider the scalar gradient (noted $|\nabla S|$). A high gradient means

(a) (b)

Figure 1.13 – *(a) An environment map (left) and its computed mipmaps, one for each roughness value, resulting in slow decrease in the represented frequency. (b) An irradiance map, only low frequency information are kept. All images are from Lagarde [2012].*

that light has probably encountered an interface, implying that light may be reflected, which requires to use a BRDF. On the contrary, a low gradient means that the element is inside a medium and thus light will most likely be scattered, according to the appropriate phase function. In practice, the local medium property should also be taken into account. Thus, the integral in the gain term (Equation 1.6) can also be discretized according to the number of light sources:

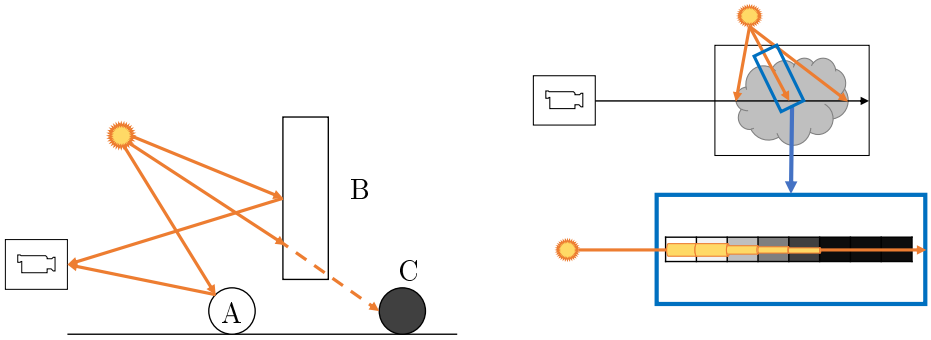
$$L(p, \vec{\omega}_{out}) = Q_e(p) + L_a(p) + \sum_{i=0}^{\#lights} I(p, \vec{\omega}_{out}, \vec{\omega}_i) \cdot L_i(p, \vec{\omega}_i)$$

with I being:

$$I = \begin{cases} f_r(p, \vec{\omega}_{out}, \vec{\omega}_i) \cdot (\vec{\omega}_i \cdot \vec{n}) & \text{if } |\nabla S| > \epsilon \\ \mathcal{P}(p, \vec{\omega}_{out}, \vec{\omega}_i) & \text{if } |\nabla S| \leq \epsilon \end{cases} \tag{1.15}$$

The aspects presented above are limited to discrete light sources (either directional, point or area). If the ambient term in the direct lighting offers a solution to approximate complex lighting environments at nearly no cost, this solution is obviously limited as it is uniform on the object and view-independent.

A popular alternative to this term is to use a precomputed lighting environment as images. By using textures, one can reproduce a mirror-like reflection of an environment, as illustrated in Figure 1.13, with only one texture access. However, a glossy or a Lambertian reflection pattern would require many texture accesses to compute an integral on the BRDF lobe to provide the correct contribution of the environment. This is equivalent to considering that each pixel of the environment map is a light source. However, this process can be precomputed to produce a set of textures corresponding to the



(a) (b)

Figure 1.14 – *(a) Shadowing in scene with completely absorbing objects and non-absorbing atmosphere. A and B are then visible by the light while C is occluded. (b) The precomputation of the visibility factor used with absorbing media (in volume).*

different material properties, as illustrated in Figure 1.13, thus producing Prefiltered Environment Maps (introduced by Kautz *et al.* [2000]). This can be efficiently implemented on modern GPU by using texture mipmap, the result is interpolated between two levels if necessary. Thus, using this Prefiltered Environment Maps, a complex lighting environment can be simulated for a low computation cost, with only one texture access instead. Furthermore, for low frequency representations, Ramamoorthi and Hanrahan [2001] propose to encode the irradiance obtained from the image into a spherical harmonics basis. An irradiance map can be converted in only 9 coefficients per color.

Shadowing and Visibility

As stated in the introduction, some objects or some parts of a volume may act as occluders for the light. This can be modeled by adding a visibility factor $v(p, \vec{\omega})$ in the lighting computation:

$$L_s(p, \vec{\omega}_{out}) = L_e(p) + L_a(p) + \sum_{i=0}^{\#lights} v(p, \vec{\omega_{in}}) f_r(p, \vec{\omega}_{out}, \vec{\omega}_i) L_i(p, \vec{\omega}_i)(\vec{\omega}_i^T \cdot \vec{n})$$

$$L_v(p, \vec{\omega}_{out}) = L_e(p) + L_a(p) + \sum_{i=0}^{\#lights} v(p, \vec{\omega_{in}}) I(p, \vec{\omega}_{out}, \vec{\omega}_i) L_i(p, \vec{\omega}_i)$$

This factor can be precomputed or computed on the fly and has a binary behavior: the element is either visible or occluded. A two-pass approach can be used for surfaces: a first pass rendering only depth information from the light source point of view, a second one from the camera point of view, using the previous information to determine the visibility factor. For a complete

overview of shadowing techniques, we refer the reader to Eisemann *et al.* [2011] and Woo and Poulin [2012].

However, shadowing techniques are limited when a complex lighting environment is used, such as precomputed environment maps. In these cases, we can not evaluate the visibility for each pixel of the environment map as this would imply too many computation. This problem can be partly addressed by Ambient Occlusion (AO). The idea of ambient occlusion is to compute a local occlusion factor, depending on the surrounding elements. This factor is computed by integrating an occlusion function over a hemisphere centered on the normal at the current point, as illustrated in Equation 1.16. Several occlusion functions have been proposed over the years (e.g., Bavoil *et al.* [2008], Loos and Sloan [2010], McGuire *et al.* [2011]) to benefit for the increasing power of GPU. An advantage of this approach is that it can be view-dependent and offers a good trade-off between image quality and computation cost compared to ray-traced approaches.

$$AO(p, \vec{\omega}_{out}) = \int_{\Omega} \left(\int_{0}^{radius} AOFunction(p + \vec{u}^T \cdot \vec{\omega}, \vec{\omega}_{out}, \vec{\omega}) du \right) d\omega \quad (1.16)$$

Ambient occlusion can be easily extended to volumes by integrating over a complete sphere otherwise. Indeed, Ritschel [2007], and later Kronander *et al.* [2012], proposed to precompute a visibility factor $v(p, \vec{\omega})$ that accounts for the whole volume. This technique consists in precomputing for each voxel the total absorption for a discrete set of directions. For each voxel, a finite set of rays are cast and the absorption is accumulated using the blending operator described in Equation 1.14, which is equivalent to evaluate Equation 1.17

$$v(p, \vec{\omega}_{out}) = \tau(p, \infty) = e^{-\int_{p}^{\infty} K_t(u) du} \quad (1.17)$$

The resulting information is projected on a Spherical Harmonics basis (directional basis) and is later used during the ray-casting to determine the visibility factor of the lights. Thus, by using precomputation, this approach provides a trade-off between accurate, but slow, ray-traced shadows, and fast, but less accurate, shadows obtained using shadow map techniques.

Global Illumination Techniques

In this section, we give an brief overview of global illumination algorithms for both surfaces and volumes. For a more elaborate review on these algorithms, we refer the reader to several other documents: Cerezo *et al.* [2005] and Jönsson *et al.* [2013] for participating media, Dutre *et al.* [2006], Ritschel *et al.* [2012] and Akenine-Möller *et al.* [2018] for surfaces, with possible combination with participating media.

Ray-Tracing and Monte-Carlo Integration The technique often referred to as ray-tracing was introduced in the graphics community by Whitted [1980]. To generate an image, a ray is launched for each pixel of the camera toward the scene in order to find the closest intersection. At any intersection point, there are two possibilities:

- the surface is diffuse: shadow rays are cast toward light sources to evaluate
- the surface is reflective: another ray is shot in the reflection direction to continue the computation.

This process is repeated for as many bounces as desired, and was later extended to handle all aspects of surfaces, e.g., diffuse inter-reflections or glossy surfaces. Theoretically, to correctly evaluate the Rendering Equation 1.9, an infinite set of rays must be launched for each bounce. In practice, this process is reduced to a finite set of rays by using Monte-Carlo integration, and the set of rays is usually generated by using importance sampling with the BRDF. This process, with a recursive propagation, means that we build the path that the light follows inside the scene, and is often referred to as Path Tracing.

Photon Mapping Introduced by Jensen [1996] for surfaces, Jensen and Christensen [1998] for volumes, it consists in shooting a set of photons from the light sources. If a photon reaches an element (vertex or voxel depending on the data representation), it is stored in this element. In a second (gathering) pass, rays are cast from the camera. As these rays interact with a surface or a volume, the photons in a neighboring area are gathered to estimate the radiance. The first version presented by Jensen was biased (the converged results may not be the correct solution) and was improved by Hachisuka *et al.* [2008] to greatly reduce the bias. Like ray-tracing, the visual quality of this technique (Photon Mapping) is greatly dependent on the number of emitted photons. If the number is too small, many artifacts may appear (an example is presented in Dufay [2017]).

Discrete Representation Some techniques have been proposed to solve Equations 1.9 or 1.8 using discrete representations. The general idea of these methods is to decompose the scene into small elements and the equations using these elements as primitives. Usually, the resolution uses either a Finite Element Method (FEM) or a Finite Difference Method (FDM) to solve the rendering equation used.

Among these, an example is Radiosity, introduced by Goral *et al.* [1984], and presented in details in Sillion and Puech [1994]. This technique uses geometrical patches to decompose a surface-based scene. For each patch, the radiosity value (the physical quantity) is precomputed and stored in the patches. The final result is then computed using a finite element representation. This

technique was later extended to using voxels to decompose the surfaces instead of patches Thiedemann *et al.* [2011]. Note that even if it was designed for surfaces, the physical model that inspired Radiosity lies in heat transfer theory, and thus, is close to the Radiative Transfer Equation (Equation 1.4). Note that this method may be faster than ray-tracing approaches, but the precomputation requires a significant memory overhead.

Another example is proposed by Zhang and Ma [2013]. They solve a diffusion equation based on the RTE (Equation 1.4), instead of Equation 1.8, by using a finite difference approach. To achieve competitive computing time, the model is simplified in regard to the RTE and a trade-off is required as single scattering and multiple scattering are computed separately. Indeed, as stated by Koerner *et al.* [2018], using an implicit representation to solve the RTE is risky as many solutions are either unstable or do not converge toward the correct solution.

1.4 Illustrative Stylisation

While current real-time and offline rendering techniques are convincing for most objects, some models suffer from lack of real landmarks and thus, from grasp of the shape. To address this issue, another type of rendering emerged in Computer Graphics: Expressive Rendering (Gooch and Gooch [2001] and Strothotte and Schlechtweg [2002]). Its purpose is to convey information about the object that we visualize, regardless of physical accuracy. This information corresponds to the message that one wishes to transmit and can be, for example, a geometrical information for an object, a temperature information for a flow, a specific material among others, etc. This message is then visualized using an adapted style, ranging from a color palette to rendering in the form of lines. Thus, expressive rendering is sometimes called Non-Photorealistic Rendering (NPR), as some styles may require to shift away from physically plausible rendering techniques.

Expressive rendering techniques originate from artistic techniques, such as painting or drawing, but also techniques based on human perception. Cole *et al.* [2008] illustrate this for line rendering with a study of the different lines that artists draw to represent the same object. Thus, when these artists must convey a similar information, the features they choose to highlight are not necessarily the same. Figure 1.15 illustrates a multitude of possible representations of the same object, where each artist has his own style to represent this object.

Thus, expressive rendering is a vast field where we can find stylization techniques, as well as illustration or geometric description. In this thesis, we focus on the geometric description of an object to highlight important geometric information. To do so, we study two types of algorithms: the algorithms mod-

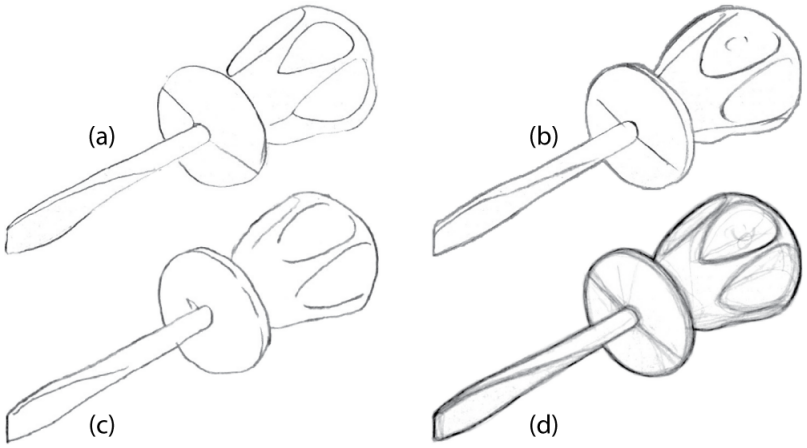


Figure 1.15 – *Examples of drawings from the same point of view (Cole et al. [2008]). (a)(b)(c) Three drawings by three artists. (d) Superposition of 14 drawings.*

ifying the lighting calculations and the algorithms producing line renderings (initiated by Saito and Takahashi [1990]). Thus, we first present examples of shading-based algorithms (interacting with lighting simulation), then we present some examples of line rendering algorithms.

Note that, even if most of the presented techniques were proposed for surfaces, they can be easily extended to volumes using isosurface rendering or, in some cases, with a combination of volumes and isosurfaces.

1.4.1 Examples of Shading-based Approaches

Gooch Shading

Gooch Shading was introduced by Gooch *et al.* [1998], inspired by the Phong BRDF (Phong [1975]), to propose a technique relying on color palettes, thus using our ability to interpret color temperatures (see Figure 1.16(a)) to transmit geometric information. The proposed model should replace the computation of illumination defined by the equation of the rendering (Equation 1.9) such as:

$$L_r(\vec{l}) = \left(\frac{1 + (\vec{n}^T \cdot \vec{l})}{2}\right) k_{cool} + \left(1 - \frac{1 + (\vec{n}^T \cdot \vec{l})}{2}\right) k_{warm} \tag{1.18}$$

where k_{cool} corresponds to a cold color and k_{warm} a warm one.

Similar techniques have been used in the form of textures, in particular with the X-Toon shading by Barla *et al.* [2006], which is a generalization of Gooch Shading with a texture, or the Lit Sphere by Sloan *et al.* [2001], which consists of using directional textures corresponding to a pre-computed illumination.

Curvature-based Shading

We can now focus on the curvature of a surface, obtained by derivation of normals. The curvature information used is based on the principal curvatures, κ_1 and κ_2, and the mean curvature $\kappa_{mean} = \frac{\kappa_1+\kappa_2}{2}$.

One of the techniques based on curvature is the Mean Curvature Shading, introduced by Kindlmann *et al.* [2003]. It consists in using curvature information to modify the rendering in order to highlight concavities, convexities or planar areas (see Figure 1.16(b)). Kindlmann *et al.* [2003] use textures whose content is associated with different curvature values. In this method, the curvatures are calculated directly in object space, which makes it possible to have temporal coherence in the highlighted information. Note that this technique was proposed initially for volume rendering, using isosurfaces.

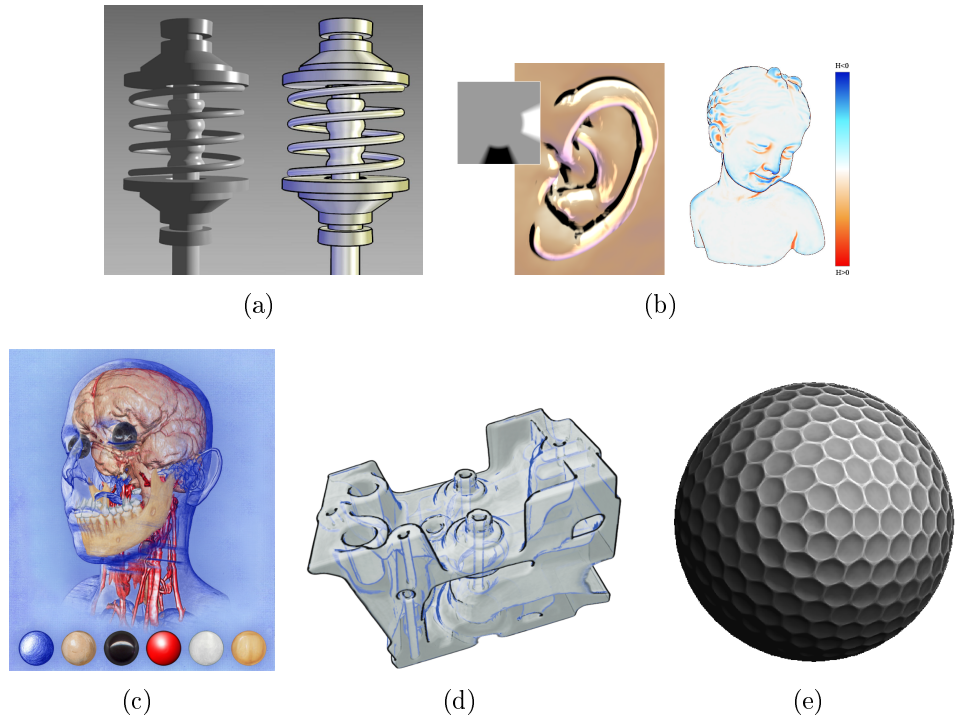
Bruckner and Gröller [2007] introduce *Style Transfer Function*, a similar approach for volume expressive rendering. They use a curvature-based transfer function for opacity combined with Lit Sphere (Sloan *et al.* [2001]) for color stylization, as shown in Figure 1.16(c). Rautek *et al.* [2007] propose to combine this kind of approach with semantic to control the stylization (see Figure 1.16(d)).

This method is used by Vergne *et al.* [2008] for the *Apparent Relief* technique by combining object space and screen space information, which allows them to take into account the point of view and the observation distance to adapt the information transmitted. The software ZBrush©proposes to use the technique of Lit Spheres (Sloan *et al.* [2001]) by indexing them on the values of curvature of the object, one for concavities and one for convexities.

Another approach using the curvature is *Light Warping* by Vergne *et al.* [2009], which consists in shifting the normals of a surface according to its curvature, and thus deforming the reflections. This deformation modifies the perception that one has of the surface by accentuating convexities and by attenuating concavities. This perceptual phenomenon is described by Fleming *et al.* [2004]. This approach was later generalized by Vergne *et al.* [2010] to work with all types of light sources (point, area, environment map...). The *Radiance Scaling* (see Figure 1.16(e)) consists in adding a scaling factor (σ) to the BRDF term $f_r(p, \vec{\omega}_{out}, \vec{\omega}_{in})$ in the rendering equation (Equation 1.9) such as it becomes:

$$f_r(\vec{l}, \vec{v})\sigma(\vec{v}, \vec{l}, \kappa) \tag{1.19}$$

The function $\sigma(\vec{v}, \vec{l}, \kappa)$ is a weighting function, which allows one to modulate the illumination in order to intensify it at convexities and to attenuate it at concavities.



(a) (b) (c) (d) (e)

Figure 1.16 – *(a) Phong Shading on the left, Gooch Shading on the right (Gooch et al. [1998]). (b) Mean curvature shading: on the left with a texture (Kindlmann et al. [2003]), on the right with a color palette (Vergne et al. [2009]). (c) Style Transfer Function (Bruckner and Gröller [2007]). (d) Using semantic to control style (Rautek et al. [2007]). (e) Radiance Scaling (Vergne et al. [2010]).*

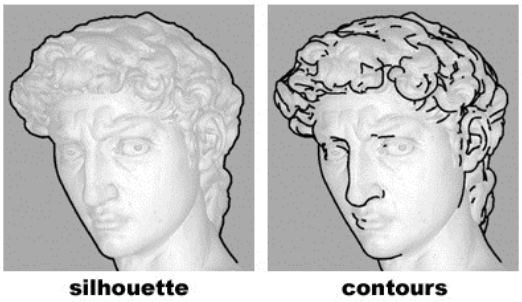


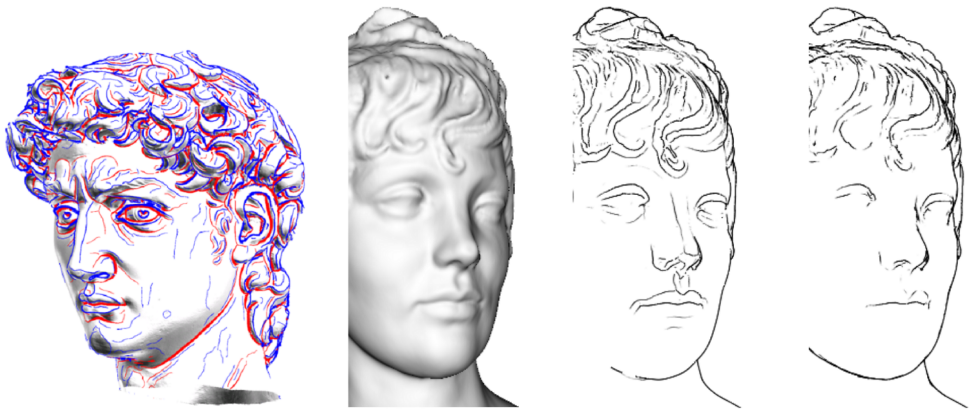
Figure 1.17 – *Silhouettes and Occluding Contours. Images from DeCarlo et al. [2003].*

1.4.2 Examples of Line Rendering

Silhouettes and Contours

Among the various line-rendering algorithms, the simplest, in term of geometry analysis, are silhouettes and occluding contours, as these two techniques require only information of depth and normals of a surface, that is, to the orders zero and one.

Silhouettes correspond to spatial limits of an object. This technique there-



(a) (b)

Figure 1.18 – *(a) An example of ridges (blue) and valleys (red), image from Ohtake et al. [2004]. (b) A shaded view on the left, Ridges and Valleys on the center, Apparent Ridges on the right, images from Judd et al. [2007].*

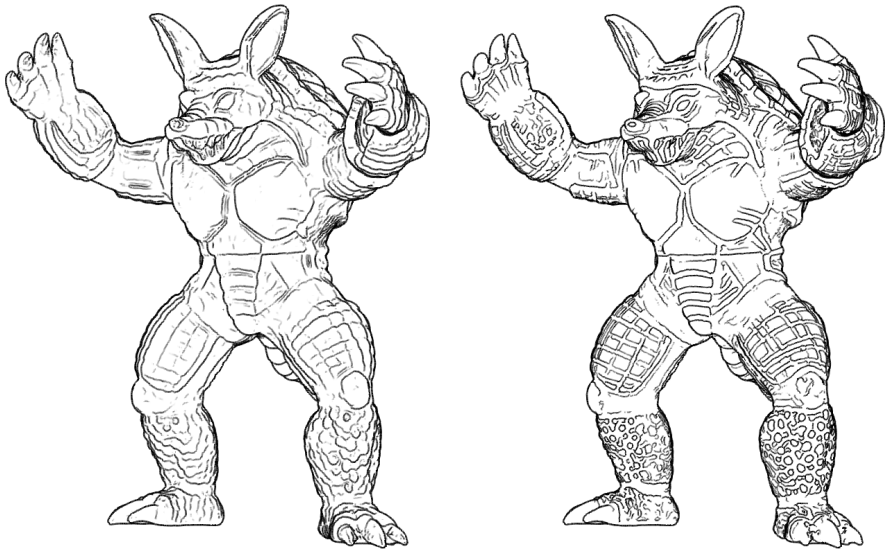
fore offers a minimalist rendering by limiting the conveyed message to border information.

Occluding contours complement silhouettes since they allow one to convey, in addition to silhouettes, simple geometric information about the object. They correspond to points where the normal of a surface is perpendicular to the point of view. An example of these two techniques is shown in Figure 1.17.

Ridges and Valleys

One of the techniques used to complete the information conveyed by occluding contours is the Ridges and Valleys, introduced by Interrante *et al.* [1995] and improved by Ohtake *et al.* [2004]. This technique consists in identifying the points of a surface where the absolute curvature is maximum. These points are thus determined by using information of curvature and curvature variations. Using the sign of curvature, it is possible to discriminate between ridges (convexities) and valleys (concavities). This technique thus makes it possible to communicate relief information, as computations are done directly on the object, independently of the point of view (see Figure 1.18(a)).

This technique can produce renderings that are difficult to understand because of the quantity, sometimes too important, of highlighted features. Judd *et al.* [2007] therefore propose to use only a subset of ridges and valleys. This subset defines Apparent Ridges, where points of interest are identified thanks to the use of an apparent curvature, which depends on the point of view. An example of Apparent Ridges is shown in Figure 1.18(b).



(a) (b)

Figure 1.19 – *Line stylization of the Stanford Armadillo model with: (a) Suggestive Contours, (b) Demarcating Curves. Images are from Kolomenkin et al. [2008].*

Suggestive Contours

DeCarlo *et al.* [2003] have chosen, with the Suggestive Contours (Figure 1.19(a)), another approach to convey geometric information: rather than focusing on the maximum curvature of a surface, they chose to highlight inflection points of the surface (where the sign of the curvature changes). They propose to highlight the points that would be considered as occluding contours if the point of view was slightly shifted, thus identifying the so-called "suggestive" contours. This definition corresponds to the points where the mean curvature is equal to 0, and its directional derivative (in the direction of the point of view) is positive. An extension can be found in DeCarlo and Rusinkiewicz [2007] by using a combination of suggestive contours and suggestive highlights, obtained from inflection points with a directional derivative being negative. Finally, Kolomenkin *et al.* [2008] propose to use Demarcating Curves (Figure 1.19(b)) that are lines defined by all inflection points, regardless of the point of view.

1.5 Visualization with Open Inventor

As the work presented in this manuscript is conducted within the toolkit Open Inventor, it is important to describe it. Open Inventor is an SDK created in 1988 by Silicon Graphics (SGI) under the name IRIS Inventor. It is now developed and maintained by Thermo Fisher Scientific. This SDK aims at providing visualizing solutions for CAD applications, medical analysis and oil & gas exploration. As these fields interact with both surface and volume data,

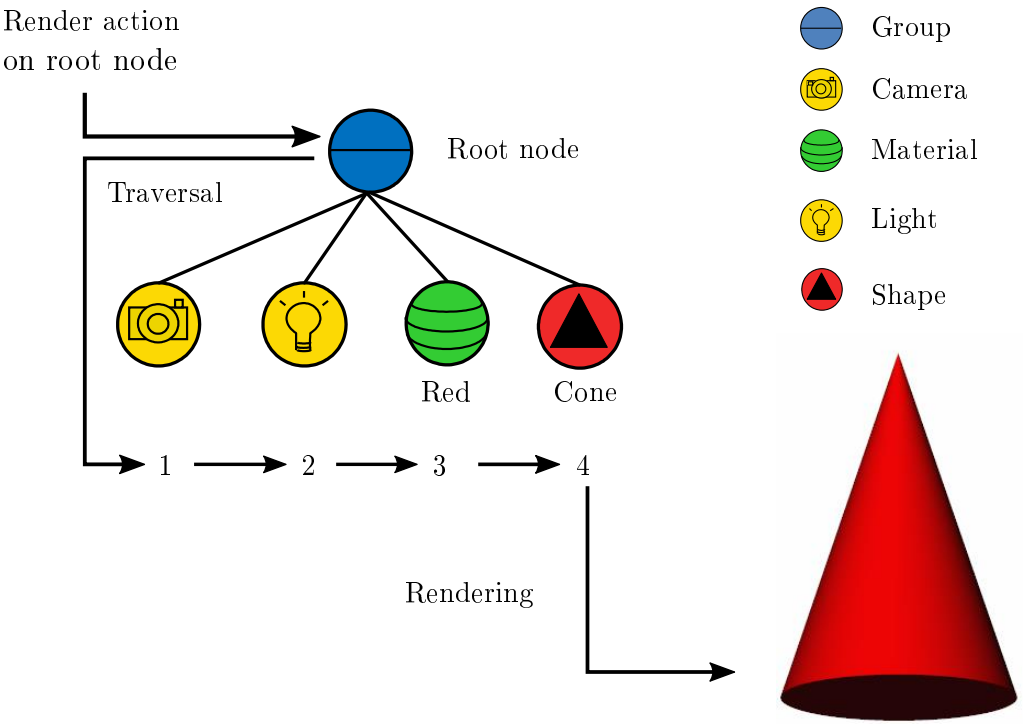


Figure 1.20 – *A scene graph displaying a red cone. A render action is performed on the root node, starting a traversal: (1) the camera node sets its properties on the state, then (2) the light node adds its properties, then (3) the material also sets the color properties and finally (4) the shape node triggers the execution of the Graphics Pipeline with the corresponding mesh and the properties accumulated on the state.*

Open Inventor handles both cases. A brief overview of the current capabilites and limitations of Open Inventor is presented here. For a detailed presentation of the features, we refer the reader to Wernecke [1994a], Wernecke [1994b] and the Open Inventor website (Inventor). Note that all the functionalities presented here are directly available in Open Inventor. As Open Inventor is an SDK, it offers possibilities for the user to implement custom features and sylization techniques.

1.5.1 Scene Graph

Open Inventor introduced the notion of scene graph (Strauss [1993]) to define a scene using nodes: shape nodes, property nodes, light nodes, etc. A scene graph offers a logical view of the scene, as illustrated in Figure 1.20. This logical view allows the user to easily build and interact with the appropriate scene.

To convert this scene graph into an image, a traversal is performed at run

time, in a depth-first order, to accumulate and execute the appropriate actions and rendering commands. Actions are the core of how Open Inventor works. Indeed, actions handle the interaction with a scene graph so that Open Inventor can interpret it and display the corresponding scene. The main actions are:

- Render Action: this action triggers the rendering of the scene.
- Bounding Box Action: this action triggers the computation of the bounding box of the scene.
- Pick Action: this action allows one to select on the screen, thanks to the mouse, a point of the scene.

An action toggles the traversal of the graph starting by the node onto which the action is performed. During the traversal of the graph, the property nodes are stored in the traversal state, which qualifies the current properties of the scene at any time of the traversal. These modifications are done thanks to elements: each node likely to modify the state possesses an element in order to transmit the properties of the nodes toward the state. Thus, for a Render Action for example, all the nodes modifying properties of rendering are updated on the state: transformations (e.g., translation, rotation), material properties, *etc.*...

As an example, Figure 1.20 displays a red cone. First, a bounding box action is performed to set the viewing volume according to the scene. Then, a Render Action is executed on the node at the root of the graph in order to toggle the traversal. During traversal, the following nodes are encountered:

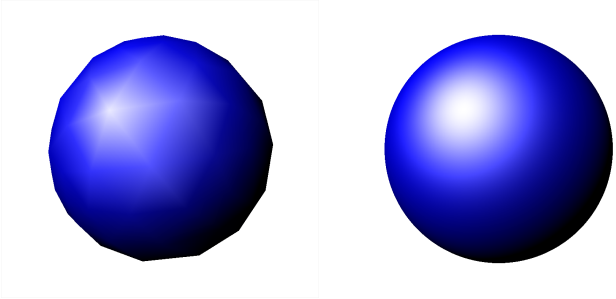
- A camera node, which puts on the state the properties of the camera.
- A light node, which puts on the state the properties of the light source.
- A material node, which places on the state the material properties (here the red color).
- A shape node, which toggles the drawing of a cone with the properties previously placed on the state.

1.5.2 Surfaces

In this section, we present the key features implemented in Open Inventor to visualize surfaces. First, we present the surface representations handled by Open Inventor, then we present its capabilities regarding transparent surfaces and finally we present the lighting and stylization techniques available in Open Inventor.

Representation and Rendering Paradigm

As it is a common representation, Open Inventor handles polygonal meshes. It also handles some implicit surfaces, represented only by a set of parameters, e.g., for sphere: its center and it radius. These surfaces are rendered using ray-marching to solve a ray intersection equation. The available shapes (also



(a) (b)

Figure 1.21 – *Comparison between two sphere representations: (a) defined by a non-accurate mesh, (b) defined as a algebraic shape. The former is less accurate but has a rendering cost much lower than the latter, offering a perfectly accurate shape.*

called Algebraic Shapes) are spheres, cylinders and cones. The main advantage of these representations is to provide perfectly smooth shapes while avoiding to use a very fine mesh model. A comparison is shown in Figure 1.21.

Open Inventor also supports parametric surfaces, such as Non-Uniform Rational Basis Splines (NURBS), which are often used for CAD applications, in particular for the design of mechanical parts.

Transparent Surfaces

Applications like CAD often require an in-depth inspection of an object. When this object is defined by surfaces, it is important to see through them in order to visualize all the parts of the object. In Open Inventor, it is done by making these surfaces as semi-transparent using a material node and blending them using Alpha Blending.

However, handling correctly transparent surfaces is not straightforward, in particular because we must take care of the rendering order of the objects. As presented in Chapter 2, several state-of-the-art algorithms are implemented in Open Inventor.

Lighting and Stylization

Direct Lighting Illumination in Open Inventor is limited to direct lighting, with two approaches:

- Base color: the lighting just takes into account the diffuse component of the material, regardless of any direction.
- Using a BRDF: the lighting is computed according to the Blinn-Phong BRDF model (Blinn [1977]).

If these two features are sufficient for some applications in which visualization is mostly illustrative, it is limited for the applications that require a more physically accurate representation.

Also, there is currently no built-in solution to handle more than point lights, directional lights and spot lights.

Shadowing Open Inventor implements two techniques for rendering shadows: Shadow Maps (introduced by Williams [1978]) and Variance Shadow Maps (Donnelly and Lauritzen [2006]). However, for surface data, it is limited to per-light shadowing, thus there is currently no built-in solution to perform Ambient Occlusion. It can however be implemented using custom nodes and custom shaders.

Expressive Rendering Currently, Open Inventor implements only few techniques to provide illustrative visualization. Most of the expressive rendering that can be done using Open Inventor are the result of custom nodes and/or custom shaders.

1.5.3 Volumes

The visualization of volume data in Open Inventor is done by the module VolumeViz, which handles data loading, formatting and rendering. First, we present the different volume rendering paradigms implemented in VolumeViz, then we present an overview of the lighting and stylization techniques.

Rendering Techniques

Most of the rendering techniques presented in Hadwiger *et al.* [2006] are implemented and available in VolumeViz. Thus, it is possible to visualize data using:

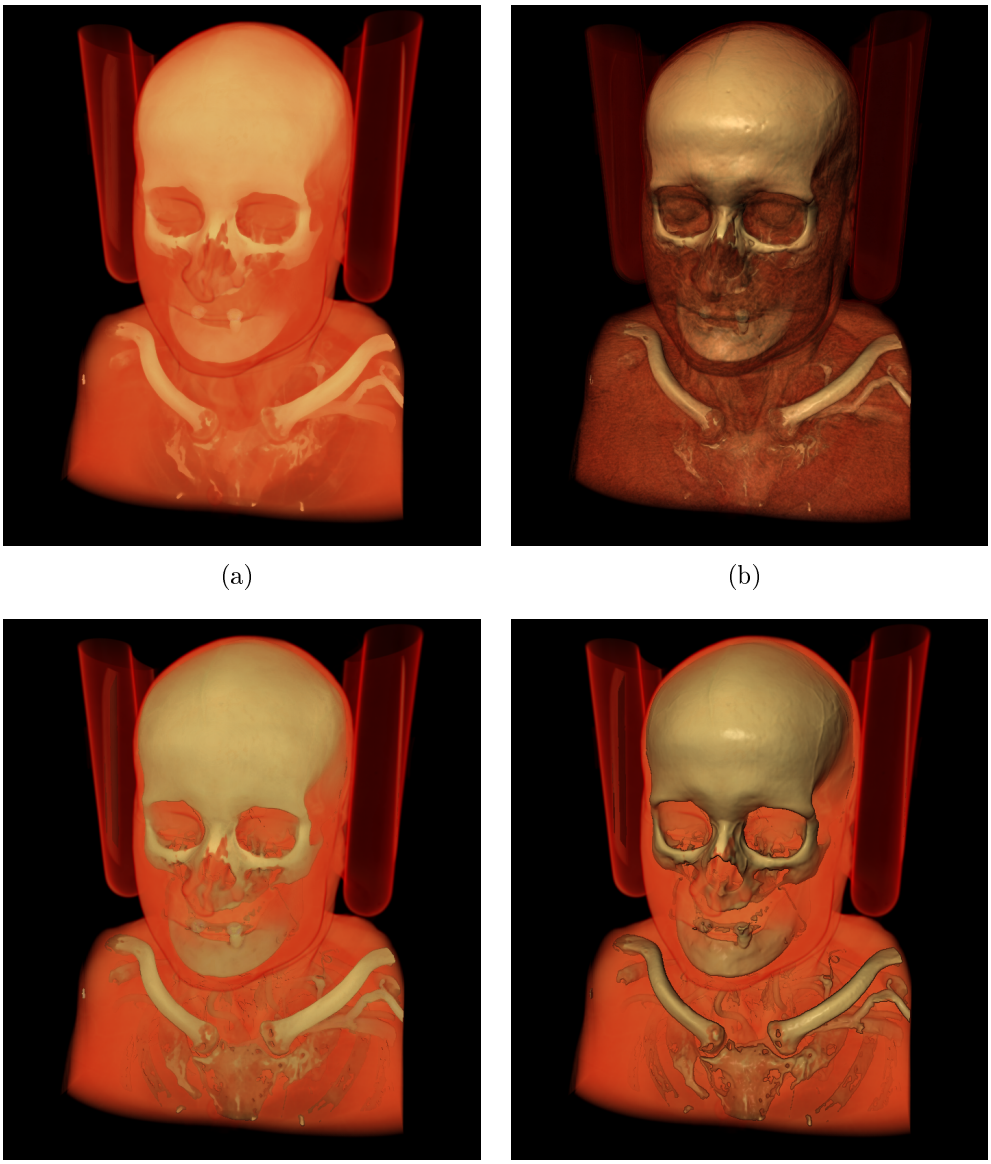
- Orthoslice: only one slice is displayed.
- Texture slicing: all slices are blended using the process presented in Section 1.3.2.
- Ray-casting: the volume is traversed using ray-marching, as presented in Section 1.3.2.
- Isosurface: extraction is done per-sample using a ray-casting pipeline.

However, there is, to this day, no built-in rendering techniques providing global illumination features in VolumeViz.

Lighting and Stylization

VolumeViz currently uses the Emission-Absorption (Figure 1.22(a)) approximation to solve the Volume Rendering Integral (Equation 1.8). The mapping from the scalar value to the associated absorption coefficient and emissive color is done with a transfer function (classification step, as evoked in Section 1.3.2):

$$TF(s) = (R, G, B, \alpha)$$



(a) (b)

(c) (d)

Figure 1.22 – *Different lighting possibilities of VolumeViz: (a) using the Emission-Absorption model, (b) with per-sample direct lighting, (c) Emission-Absorption with Ambient Occlusion, (d) deferred direct lighting with Ambient Occlusion.*

Direct Lighting In term of lighting simulation, the VolumeViz pipeline handles direct lighting with a BRDF only, using the scalar gradient to determine whether the sample should be lighted or not (Figure 1.22(b)). In the latter case, the lighting simulation just outputs the emission value. Note that direct lighting in VolumeViz can be done as a deferred process (Figure 1.22(d)) a first pass does classical non-lighted volume rendering and the closest non-transparent sample per pixel is stored in a texture. Direct lighting is then

applied to these samples only. Note that the specular component of the direct lighting is handled with a material node.

Since the 9.8 version, VolumeViz also implements image-based lighting (presented in Section 1.3.3) and handles Tone-Mapping for High Dynamic Range (HDR) output images (see Reinhard *et al.* [2005] for an introduction to HDR).

Shadowing In terms of shadowing capabilities, VolmueViz embeds more options than Open Inventor for surfaces. Indeed, it implements the same shadow map algorithms (Williams [1978] and Donnelly and Lauritzen [2006]), but also Ambient Occlusion as a deferred post process (Figure 1.22(c)), performed on the first hit-point.

Since 9.8 version, VolumeViz also implements ray-traced shadows: for each sample, a ray is cast toward each light source to evaluate the visibility of this source.

1.6 Conclusion

In this chapter we have presented several aspects of light transport: how it is used in Computer Graphics to render both surfaces and volumes and how it can be modified to provide non-photorealistic renderings. We have also presented briefly Open Inventor, the SDK that we use for the work presented in the following chapters.

We illustrated that, despite its current features and the possibility to implement custom ones, Open Inventor lacks ready-to use solutions to enhance the depiction of transparent objects (surfaces or volumes). Furthermore, we also concluded that there was no satisfying solution in the literature to address the specific problem of transparent surfaces. In Chapter 2, we propose a real-time solution to this problem.

For the case of volumes, approaches to highlight specific features often rely on segmentation, more than using light transport. However, segmentation-based visualization is a tedious process and many of the known algorithms are task-specific. Furthermore, these algorithms often operate at a global scale. Thus, in Chapters 3 and 4, we explore a new metaphor using light transport, based on medical imaging use cases, to identify region of interest.

Chapter 2

Feature-extraction for Visualization of Transparent Surfaces

Transparency is a common tool for analysing complex object subtleties,e.g., Computer-Assisted Design. However, with raster-based applications, ambiguities from using classical alpha-blending are twofold: (i) a correct composition implies to render the scene back-to-front; (ii) composition of multiple non-related information introduces a bias in shape perception. The back-to-front composition issue (i) is solved by Order-Independent Transparency (OIT) methods, as presented in Section 2.1.1. However, the second one (ii) has no obvious solution.

In this chapter, we introduce customizable tools that help to work around shape perception ambiguities during alpha-blending. Our main contribution is twofold. First, we introduce a data structure to perform OIT that is suited to compute derivatives in multilayered data efficiently. Second, we also introduce a customizable stylization that modulates shading and transparency per fragment, according to its curvature and its depth, which can be adapted for various kinds of application.

This work was published and presented at the EuroGraphics Symposium on Rendering (EGSR) in 2016 at Dublin (Murray *et al.* [2016]). It is currently going through the integration process into Open Inventor.

2.1 Shape Depiction and Transparent Surface

As presented in Section 1.4, shape perception can be emphasized with expressive rendering methods. Among the large set of techniques available in this domain, we are particularly interested in shape depiction algorithms that aim at stylizing the rendering of a shape according to its features. Basically, these

features can be classified into three categories: surface orientation, surface curvature and inflection points. While the first one can be considered as supplied during rendering process, other ones should be computed during rendering in order to benefit from screen-space formulation.

Screen-space approaches for feature extraction are useful for exploring dense meshes obtained by photogrammetry using Radiance Scaling (Granier *et al.* [2012]), with applications in Cultural Heritage. We consider this as we focus on the exploration of complex models (e.g., CAD), which requires efficiently computing curvature at a scale adapted to the current viewpoint. The computation of surface properties is linked to derivative considerations: curvature is a derivative of the normals and we need a derivative of the curvature to find inflection points. Using discrete differential geometry operators, the derivative computation in screen-space involves to have access to the neighborhood of a point, i.e., a fragment located at a pixel position. For opaque objects, depth-test operation is enough to eliminate unnecessary fragments. Thus, with one fragment per pixel, finding neighbor information is straightforward.

For transparent objects, a pixel may contain multiple fragments. Since a differential operator must be applied per fragment to perform shape depiction locally, the complexity of using such an operator is overweighted by the complexity of finding neighbor information. In order to understand the problem of combining the two approaches (shape depiction and transparent surfaces), we first present how transparent surfaces are handled in a raster-based application. We then discuss about the difficulties of using shape depiction with the structure required to render transparent surfaces.

2.1.1 Rendering Transparent Surfaces

When rendering of a surface, the geometry will first be sent to GPU by vertex before being assembled into primitives (see the Graphics Pipeline in Section 1.3.1). During rasterization, these primitives will be converted into fragments, as they are projected on camera pixels. Shading and lighting is computed for each generated fragment, whose composition constitutes the pixels of the final image. To display a scene, this process is applied to each object in the scene.

This process is straightforward for opaque surfaces as only closest surfaces are kept to generate the image. However, transparent surfaces require to deal with all surfaces, thus they require either to blend them directly or to rasterize all of them to keep their fragments. The former is directly done using Alpha Blending, however it requires depth-ordered surfaces. The latter requires a dedicated data structure to handle the fragments.

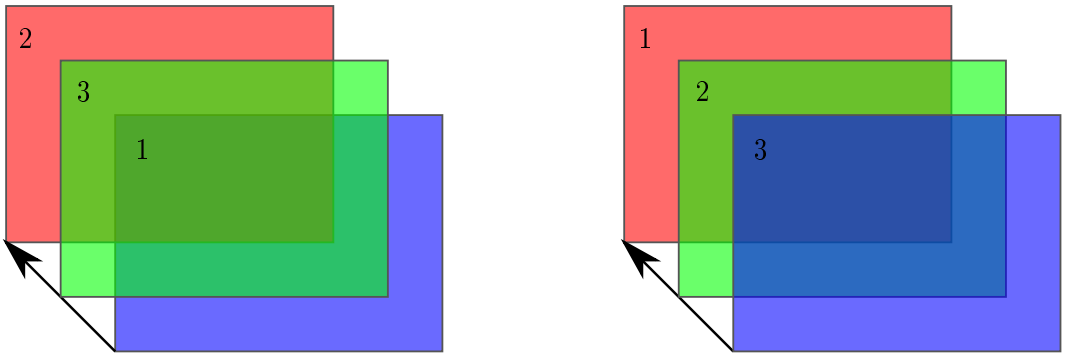


Figure 2.1 – *Importance of the rendering order with two examples of rendering order, the number on each plane corresponding to this order. On the left, objects have not been displayed in the order corresponding to their depth, leading to an inconsistent transparency. On the right, the objects have been displayed in the order corresponding to their depth.*

Alpha-Blending and Rendering Order

In the case where one wishes to use transparency, each fragment receives an opacity value, commonly called α, ranging from 0 for a totally transparent surface to 1 for an opaque surface. This opacity value is then used for the composition by using Alpha Blending, as introduced in Section 1.3.2. The composition is performed from the farthest fragment to the nearest fragment to the camera, which corresponds to the OVER operator by Porter and Duff [1984], such as:

$$RGB_d = \alpha_s RGB_s + (1 - \alpha_s) RGB_d \tag{2.1}$$

In this function, RGB_s is the RGB triplet of the current fragment, the source fragment, and RGB_d is the RGB triplet of the destination fragment.

Since this equation is not commutative, the order in which objects are displayed has a direct influence on the composition. As shown in Figure 2.1, if an object is displayed before the one that should be located behind it, or if two objects intersect, the final rendering will be distorted because the object already displayed will be considered as being the furthest during composition.

This limitation can be addressed by determining, beforehand, in which order objects must be displayed. This way, we ensure that depth order will be respected. However, the upstream processing of intersections requires heavy calculations, which can considerably reduce the performance of an application. An alternative is to move this problem from the geometric calculation step to the fragment composition step. It is then possible to get rid of the order in which the objects of the scene are displayed. The techniques that address this issue are called the Order-Independent Transparency (OIT) techniques, since Everitt [2001].

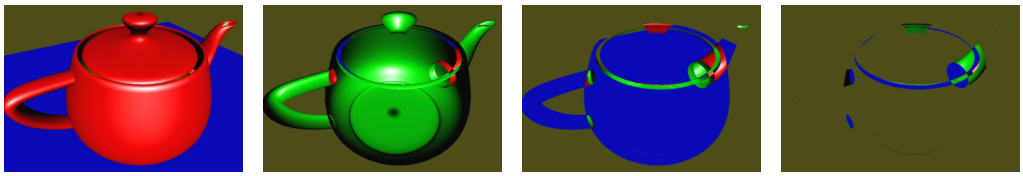


Figure 2.2 – *Depth Peeling with 4 layers (Everitt [2001]): first layer on the left, fourth on the right.*

Order-Independent Transparency

It is possible to get rid of the rendering order by using commutative operators (multiplication only $RGB_d = RGB_s * RGB_d$ or addition only $RGB_d = RGB_s + RGB_d$), however these operators do not take into account the values of α fragments. McGuire and Mara [2016] propose, as an alternative, to weigh the value of α according to the depth of a fragment during composition, however this method is more suitable for fluids than surfaces. We will therefore focus on methods that have been developed to deal with surfaces when we want to use non-commutative operators.

Depth-Peeling Introduced by Everitt [2001], it consists in splitting the scene to be displayed into a predefined number n of layers. To do this, we will display the scene n times, where each iteration generates a layer, according to the previously generated layers. The first layer corresponds to the fragments (one per pixel) closest to the camera, then rendering is performed again by eliminating the fragments previously stored to generate the second layer. This process is then repeated to have the desired n layers. Figure 2.2 illustrates how Depth Peeling operates with 4 layers. At each step, it is necessary to store the depth of the fragments to be able to eliminate the fragments having a lower depth. The layers are then blended with Alpha Blending (Equation 2.1) to obtain the final image.

This method was subsequently improved by Bavoil and Myers [2008] with the Dual Depth Peeling, which consists in decomposing the Depth Peeling into two parts: one done from the camera and one from a virtual camera located on the opposite side of the scene. However, the necessary number of iterations remains expensive and therefore remains a penalizing point of this method.

A-Buffer The A-Buffer technique was introduced by Carpenter [1984]. It consists in storing the fragments, generated per pixel during rasterization, in a linked-list, whatever the order in which they were generated. This list is then sorted by fragment depth to perform the composition of the fragments, carried out from the farthest to the nearest. This process is represented in Figure 2.3.

As all the fragments generated for each pixel are stored, the A-Buffer can lead to a saturation of the memory in the case of complex scenes, and therefore

to empty areas in the image where the fragments could no longer be inserted.

If the technique was introduced in 1984, the first functional implementations on GPU are recent, starting with the method of Yang *et al.* [2010]. Indeed the method of Carpenter [1984] is limited by the parallel architecture of GPUs. Because fragments are handled simultaneously, they may be competing to be inserted in the same place, at the same time, in the fragment list. In this case, one of the fragments overwrites the other, which will then be inserted at the wrong place (or discarded), causing rendering artifacts. Thus using atomic operations on GPUs (thread-safe read/write operations), Yang *et al.* [2010] propose an implementation that performs the insertion atomically, ensuring that only one fragment can be inserted at a time.

k-Buffer The k-Buffer is a technique in-between Depth Peeling and A-Buffer. Introduced by Callahan [2005], this technique consists in building a sorted list of fragments (like A-Buffer) with a fixed size (like Depth Peeling). Indeed, building a k-Buffer consists in keeping the k first layers of the scene, thus the k nearest fragments. To do so, fragments are inserted in the list with a depth-sorted insertion. After insertion, we have virtually the same output as with Depth Peeling (k nearest "images"), but this method requires only theoretically only one rendering pass. Due to current hardware limitations, the most used version is the dual pass k-Buffer (Kubisch [2014]) with two rendering passes: one to create the list using fragment depth, and one to store the color of these fragments.

Thus, the k-Buffer does not have the disadvantage of A-Buffer in terms of potential memory saturation. However it has a limitation since memory is pre-allocated for the k layers, which means that even if a pixel contains only one fragment, the memory provided for the other layers will still be allocated.

Finally, this technique is subject to the same technical limitations as the A-Buffer in terms of read/write hazards and therefore requires the use of atomic operations.

2.1.2 Limitations for Shape Depiction

Shape depiction techniques aim at conveying a message based on shape properties. Some previous methods propose to modify the reflection pattern according to constraints such as geometric features, material properties or stylization. An overview of these techniques is presented in Section 1.4. If some techniques, like Radiance Scaling (Vergne *et al.* [2010]), can be used for semi-transparent surfaces with a ray-tracing paradigm, most of these techniques are initially designed for opaque surfaces when used with raster-based applications.

Combining OIT with expressive rendering methods is addressed by Hummel *et al.* [2010], with Normal-Variation Transparency. However, as they use built-in GLSL functions, neighborhood is inaccurate. Carnecky *et al.* [2013]

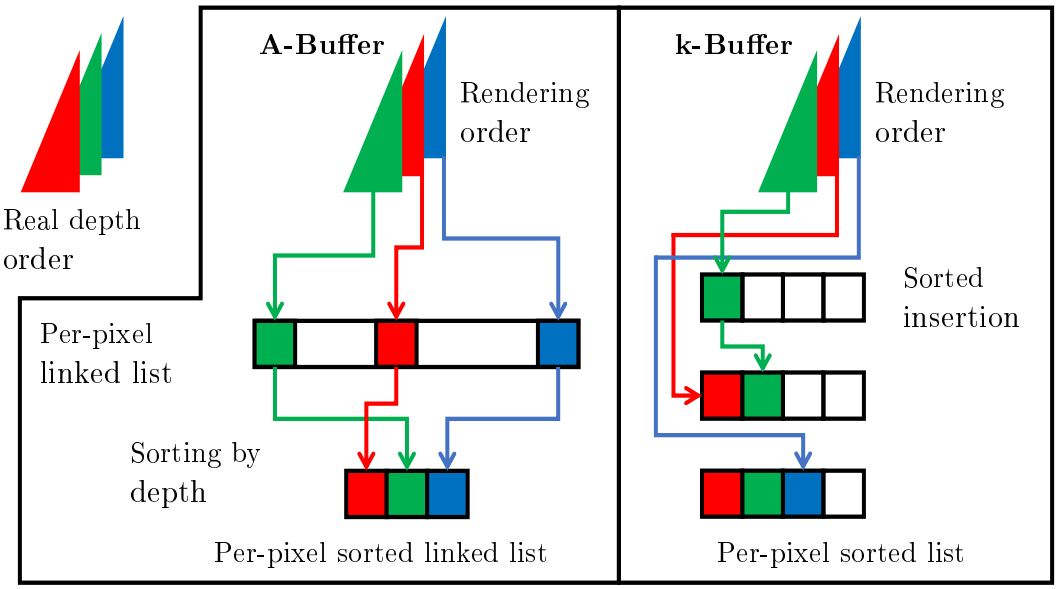


Figure 2.3 – *Principle of A-Buffer (left) and k-Buffer with 4 layers (right). Triangles on the top left corner illustrate the real depth order of the object.*

propose an altered A-Buffer, in which fragments are connected to their neighboring fragments. According to the connexity and the logical depth of a fragment, transparency is modulated such that creases are emphasized to nearly opaque, smoothly decreasing to nearly transparent otherwise. Günther *et al.* [2014] propose an enhancement with surface patches to make parts of a surface transparent if an object is behind. However, to our knowledge, there is no method that computes screen-space geometric features per fragment to modulate shading and transparency in real time. Based on a tailored data structure that allows such computation, we propose a method that provides a way to extend existing shape depiction techniques to transparent shapes.

2.2 Solution Overview

The goal of the presented method is to enhance shape depiction for transparent objects. Our process is composed of three main steps as shown in Figure 2.4:

1. Scene discretization into a screen-space representation.
2. Per fragment feature-based stylization.
3. Blending of the generated fragments.

The first step consists in capturing the scene into the Bucketed k-Buffer (Bk-Buffer). The Bk-Buffer is a discretization of the scene in such a way that fragments are organized pixelwise into three 3D data structures: Z, B and D. The roles of Z and D are nearly the same as data structures used in dual pass k-Buffer (Kubisch [2014]) to capture, respectively, depth and data. B is used to group fragments belonging to the same shape per bucket. Details of

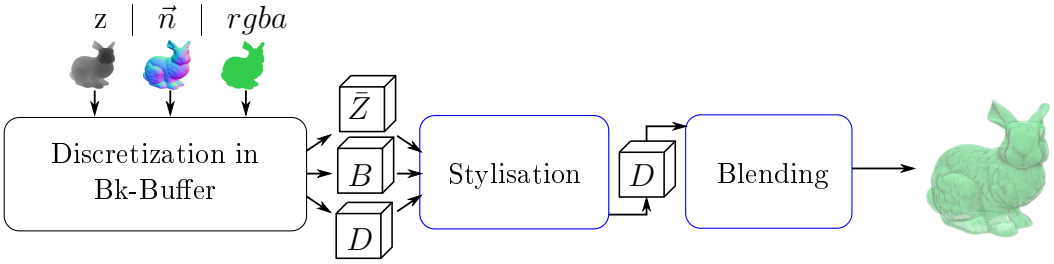


Figure 2.4 – *Pipeline used for feature-based stylization with Bucketed k-Buffer. The discretization pass fills the buffer Z, the bucket buffer B and the data buffer D with their respective information (see Section 2.3). The stylization extracts features and applies modulation on fragment's data (see Section 2.4). Finally, the blending pass consists in traditional OIT back-to-front composition.*

the Bk-Buffer's data structures are presented in Section 2.3. Note that the Bk-Buffer contains geometric and color data per fragment. The second step is a stylization process which extracts geometric features from the Bk-Buffer (i.e., based on the normals) to enhance shape depiction (i.e., by modulating shaded color). The stylization is customizable by providing user-defined transfer functions indexed by the fragment's depth and curvature (see Section 2.4). The last step consists in reordering fragments back-to-front in order to realize a correct alpha-blending.

Every stage of our system is performed in real-time on modern graphics hardware. Results are presented in Section 2.5.

2.3 Bucketed k-Buffer for efficient neighbor query

Leaving aside commutative OIT which excludes per-fragment operations, we focus on per-pixel fragment list. We seek a way to efficiently access a fragment's neighborhood. First, we introduce the concept behind our solution, that is, the elaboration of the Bucketed k-Buffer. Then, we provide several practical details about the method.

2.3.1 Definition of the structure

Our structure relies on a k-Buffer basis. Indeed, with current GPU, Depth Peeling is becoming obsolete due to the important number of rendering passes it implies compared to other techniques. A-Buffer relies on a linked list which implies unbound memory. Moreover, all the fragment informations are stored in the same buffer. For a better flexibility on data structure, the dual-pass k-Buffer (Kubisch [2014]) technique is well-suited. Indeed, by decorrelating depth and color, we can manipulate depth or color information separately. Furthermore, storing only the k nearest fragments in k layers ensures a total

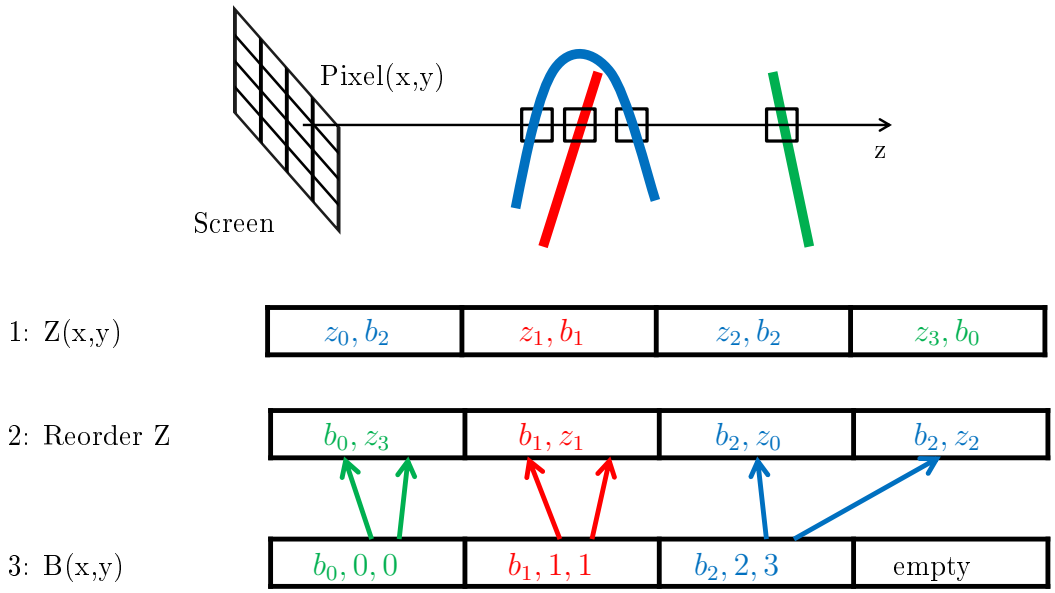


Figure 2.5 – *Description of the discretization process. In the first pass (1), Z is filled with a depth-ordered fragment list, containing depth (z) and shape ID (b) for each fragment. In the second one (2) Z is reordered by shape ID. Then (3), B is filled with bucket's information : shape ID (b), first and last fragment index (i, j).*

control over memory occupancy. Those two points are key elements for the choice of our structure.

In such a structure, a naive approach to find the best neighbor candidate is to iterate through each neighboring pixel's fragment list. However, this implies costly memory access. Vardis *et al.* [2016] use buckets to decompose a fragment list such that fragments are grouped by depth ranges. In the same way, we propose to reorganize the fragment lists so that fragments belonging to a same object are regrouped in a bucket. That way, only fragments in the corresponding bucket needs to be traversed, thus avoiding many memory accesses. Using such partition, finding a one-ring neighborhood would go down, in theory, from a complexity of $O(k^2)$ per pixel to $O(kn)$, where k is the number of fragments per pixel and n is the number of fragments in a bucket, so we should have a gain as soon as $n < k$.

The proposed algorithm shares the same concepts as the dual-pass *k-Buffer* in the sense that we use two geometry passes to store depth and data in separate buffers. However, in order to facilitate access to neighbors, our data structures differ in the data we store and we use additionnal computing passes to arrange the data according to our needs. The remainder of this section describes the pipeline to build the Bk-Buffer data structure in a sequential order. In the following, W and H denote respectively the width and the height of the rendering window.

Z insertion Similar to the dual-pass *k-Buffer* (Kubisch [2014]), we store a fragment's depth per pixel in the first geometry pass. Relying on the benefit of such approach, the n nearest fragments to the camera, $n \leq k$, are stored and ordered by depth in the buffer Z. For each fragment, we keep not only its depth (z) but also its shape ID (b) (see Figure 2.5 (1)), where b is a integer representing the identifier of the shape where the fragment is from. Thus, the Z data structure is of size $W \times H \times k$, containing pairs (z, b).

Z reordering During this pass, we reorder per-pixel fragment pairs of the Z data structure along b instead of z (see Figure 2.5 (2)). Sorting fragment pairs in this manner allows grouping fragments that belong to the same shape consecutively in order to gather them into buckets.

Bucket creation Per-pixel groups of fragments, which share the same shape ID, are implicitly defined by the reordering of Z in the previous pass. However, in order to have a direct access to the range of these groups, the B data structure is built in a computing pass. Such groups are defined as buckets: a bucket contains the shape ID (b, i.e., a bucket ID) and the indices (i, j) of the first and last fragments of a group. The bucket buffer B contains $W \times H \times k$ triplets (b, i, j) (see Figure 2.5 (3)).

Since n fragments are stored in Z during the first pass, $n \leq k$, no more than k buckets are stored per-pixel. However, storing bucket information in a table indexed by b can lead to indices that are out of range if $b > k$. Thus, B is considered as a per-pixel hash table in which each bucket is indexed by $\mathcal{H}(b)$, where $\mathcal{H}$ is a hash function defined as:

$$\mathcal{H}(b) = b \mod k$$

Using such hash function may lead to collision in the hash table. Thus, during insertion of a bucket b in buffer B, we check that the slot at index $\mathcal{H}(b)$ in the hash table is free. If the slot is unoccupied, the bucket information is stored at this index. If the slot is occupied, b is incremented until the index $\mathcal{H}(b)$ points to a free slot. Note that a free slot is always available since the number of buckets per pixel cannot be greater than k, the size of the hash table. The same incremental process is used to find a bucket, except that we check if the slot contains the right bucket. Organized in that way, B, in combination with Z, is used as an index lookup to perform efficient fragment query as shown in Figure 2.6.

Data storage To store fragment's data, a second geometry pass is performed to fill the buffer D. The fragment's data are its color (c) and normal ($\vec{n}$). Only fragments pre-selected during the first geometric pass must be stored. Thus, for each generated fragment, we perform a fragment query as shown in

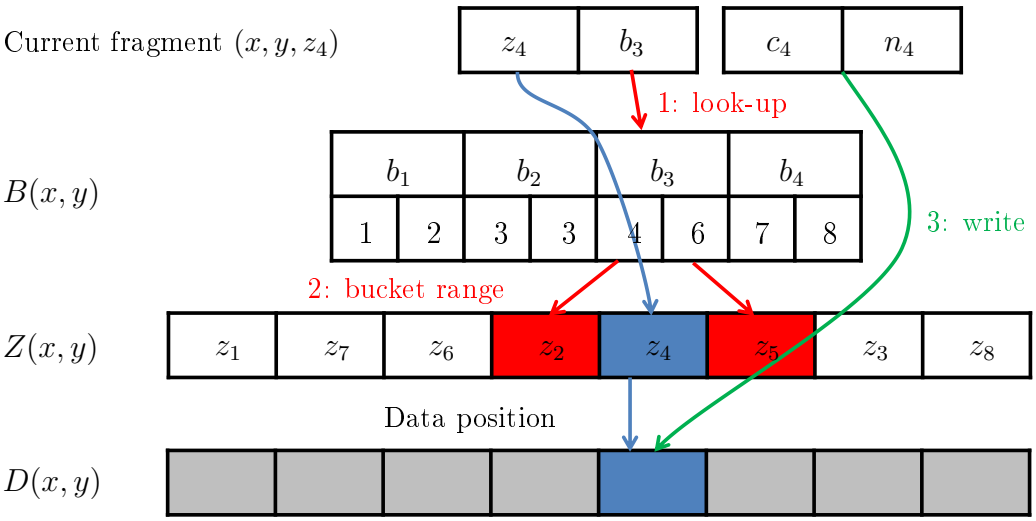


Figure 2.6 – *Fragment query for data storage pass. For a generated fragment, the shape ID is used to locate, with a linear search, the associated bucket (1) and extract its range (2). The current fragment is located within these ranges. To find its index, a depth comparison is used. Fragment's data, color and normal, are stored at this index (3). For neighbor query, the fragment (x, y, z) is used for a look-up in $B(x + dx, y + dy)$ and $Z(x + dx, y + dy)$.*

Figure 2.6. If the query is successful, data are stored to the given index, or rejected otherwise. Buffer D thus contains $W \times H \times k$ pairs (c, n).

2.3.2 Implementation details

As presented in the previous section, we manipulate three 3D structures. Each of them will be able to contain $W \times H \times k$ elements, where W and H correspond to the viewport size and k is the number of layers in the buffer.

During the first pass, the sorted-insertion by depth is done using atomic comparison. As atomic comparisons rely on a bitwise AND operator, depth must be used as Most Significant Bit (MSB), even if we store both depth and shape number. These two values are packed on 32 bits with 16 bits each, in the buffer Z. Using depth as MSB ensures that we keep the k nearest fragments.

For the construction of the buckets, depth and shape number are swapped inside Z. This way, shape number is now used as MSB instead of depth and we can easily reorder our list. After reordering, buckets are constructed as described in the previous section. Using this structure requires 32 bits per bucket : 8 bits for the first fragment position, 8 bits for the last one, and 16 bits for the shape number.

Finally, to insert the data in the buffer D, we rely on a binary search done directly inside the correct bucket. This bucket is located using the process in Figure 2.6 with a condition on shape number. Once we know the correct fragment's layer, we can store the fragment color after shading to use only 32 bits

for color. The shading relies on the Bidirectionnal Reflectance Distribution Function (BRDF) proposed by Burley and Walt Disney Animation [2012] for direct illumination and Image-Based Lighting to approximate indirect illumination. We store as well the fragment's normal, packed on 32 bits (15 bits for each x and y components and 2 for the sign of z). This results in using 64 bits per fragment for the data structure.

2.4 Feature-driven stylisation

In this section, we demonstrate how to use the Bk-Buffer for an efficient feature-driven stylization. First, we present how to find fragment's neighborhood to compute curvature per fragment. Then, an application of stylization is shown adapted from existing shape depiction techniques.

2.4.1 Feature extraction

Carnecky *et al.* [2013] propose a method to find the neighbor of a fragment. A comparison is performed between a fragment and all the fragments contained in an adjacent pixel. The comparison function (ϵ) is based on normal and depth differences and the best neighbor candidate is the fragment that obtains the minimum value. We use a similar approach with the following criteria:

$$\begin{aligned} \epsilon_n(i,j) &= 1 - \vec{n_i}^T \cdot \vec{n_j} \\ \epsilon_z(i,j) &= |z_i - z_j|^2 \\ \epsilon(i,j) &= \epsilon_n(i,j) + \epsilon_z(i,j) \end{aligned}$$

To ensure that we correctly identify the neighborhood, each fragment labeled as the best candidate does a reverse check to determine if this candidate also views the current fragment as its neighbor. If one of the neighbors is not found, we use a special value indicating a missing neighbor. Figure 2.7 illustrates the interest of this reverse check. Comparing to the work of Carnecky *et al.* [2013], a crucial point in our method is the lower complexity of this algorithm. Actually, thanks to the Bk-Buffer, fragments belonging to the same object are grouped together. Thus, as explained in the previous section, the number of pairs to test is restricted to a subset of candidates, i.e., the ones with the same shape number (ID). We access to this subset using the same process as data storage, presented in Figure 2.6, except that the query is done on the neighboring pixel's lists (buckets and fragments).

The neighbor information is used to compute screen-space curvature per fragment, such as Vergne *et al.* [2009]. In this approach, principal curvatures, κ_1 and κ_2, are extracted from surface normals. Instead of directly using the

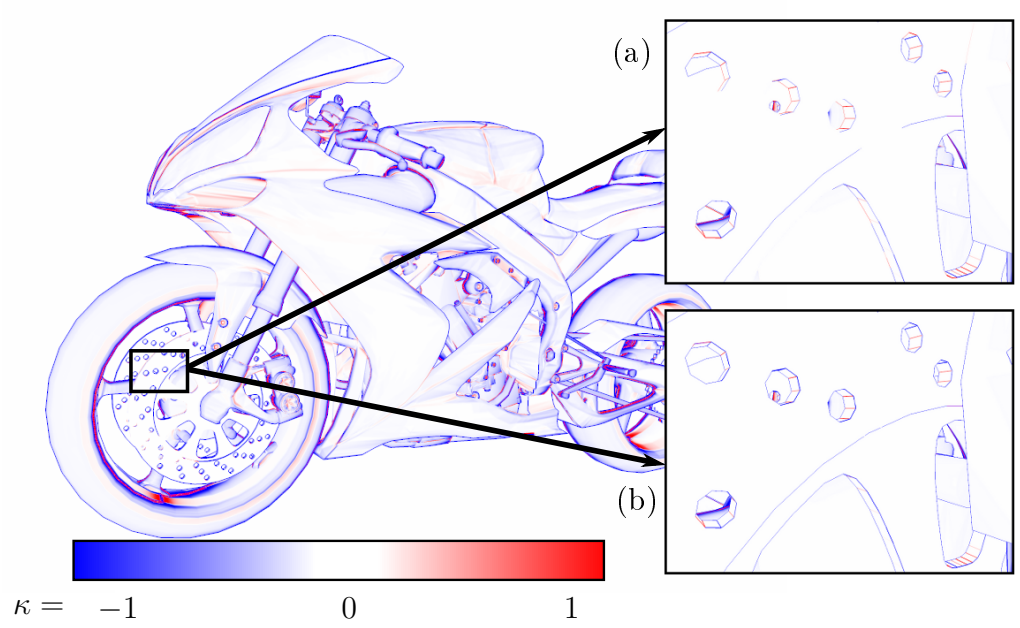


Figure 2.7 – *Importance of doing a reverse check. (a) A simple check, which results in biased results at creases. (b) A reverse check, with creases identified as such.*

mean curvature $(\kappa_1+\kappa_2)/2$, we prefer, for practical reason, the following mapping varying between -1 and 1:

$$\kappa = \tanh(\kappa_1 + \kappa_2)$$

For the rest of this paper, κ is considered as a curvature value. In our convention, -1 corresponds to a convexity and 1 to a concavity. Contrary to Vergne *et al.* [2009], our method does not suffer from an undefined behavior at the object's boundaries. By taking advantage of potential missing neighbors, we can handle those aera: the curvature of a fragment with a missing neighbor is set to $\kappa = -1$.

2.4.2 Stylization

For the stylization, our method modulates both shaded color and transparency. These modulations are controlled by transfer functions: a 1D texture for color and a 2D one for transparency.

The shaded color is modulated with a curvature-indexed transfer function, presented in Figure 2.8(a). This function is inspired by Mean Curvature Shading (Kindlmann *et al.* [2003]) and Radiance Scaling (Vergne *et al.* [2010]), with bright convexities and dark concavities. It provided convincing results for our

Figure 2.8 – *(a) The transfer function we use to modulate shading brightens convexities and darkens concavities. (b) Another example: bright curved surfaces, dark flat ones.*

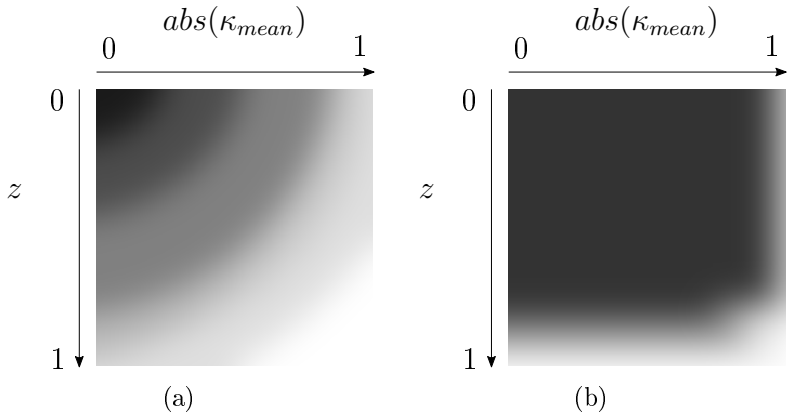
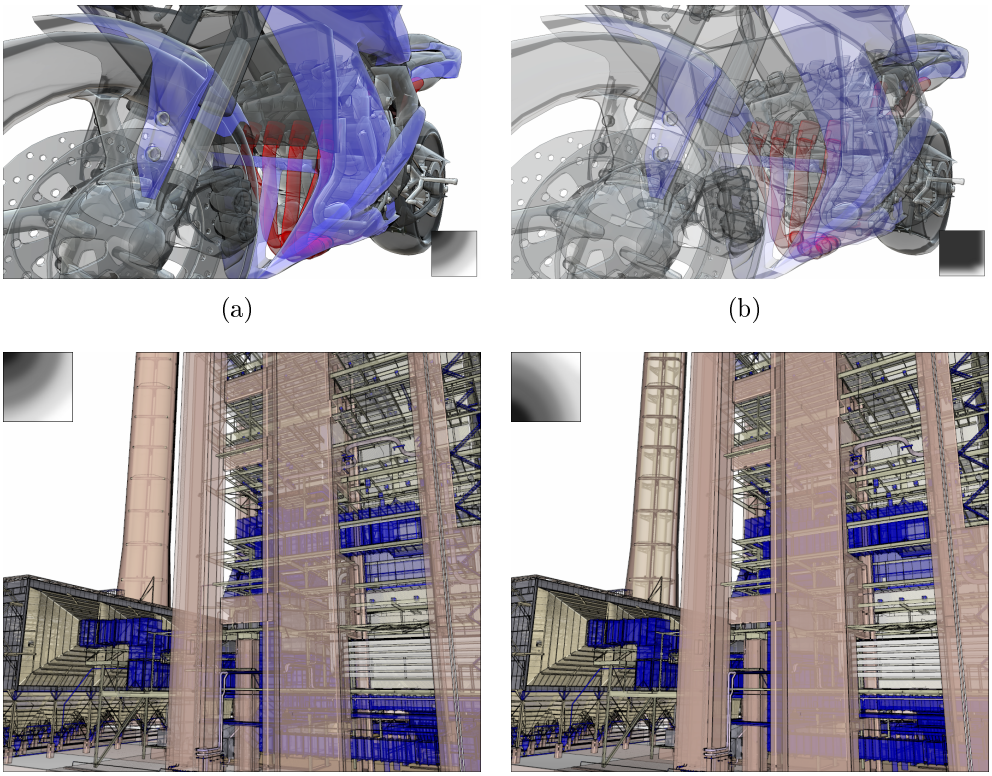


Figure 2.9 – *Example of different transfer functions. White corresponds to opaque (opacity of 1) and black to transparent (opacity of 0). Both follow the same idea: more transparency on flat and close surfaces, more opacity on curved surfaces as well as distant ones.*

application, as we wanted to modulate a physically-based shading. However, for other applications, a different transfer function could be used in our pipeline for other stylizations as shown in Figure 2.8(b).

The modulation of transparency extends the work presented by Hummel *et al.* [2010] as we use curvature instead of normal-variation considerations. Based on the idea of X-Toon (Barla *et al.* [2006]), we propose to modulate opacity with a transfer function indexed by absolute mean curvature and depth. A value of 1 corresponds to fully opaque and 0 to fully transparent. An example of transfer function can be seen in Figure 2.9. We choose to use the absolute



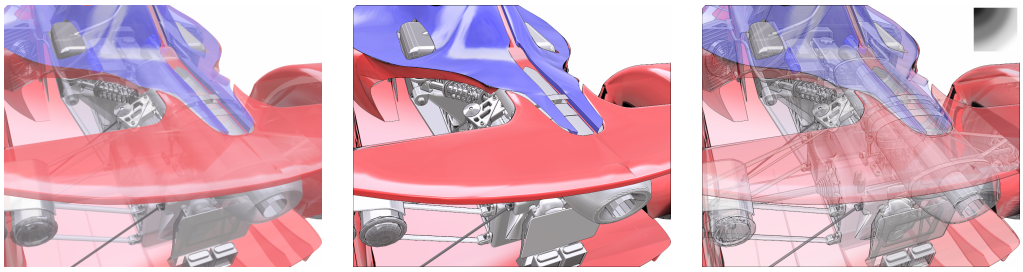
(a) (b)

(c) (d)

Figure 2.10 – *Examples of Mean Curvature Transparency. (a,b) The transfer functions used are respectively the one presented in Figure 2.9 (45 FPS versus 120 FPS without modulation). (c,d) The transfer functions are shown in the top left corner (10 FPS, versus 20 FPS without modulation). Times are obtained with a resolution of 1200 by 800 for the bike, and 1000 by 1000 pixels for the UNC Powerplant, with 16 layers for both cases.*

mean curvature so that either convexities or concavities would have the same opacity. Indeed, based on our observations, both are crucial to fully grasp shape characteristics.

Our system allows the use of any transfer function to modulate opacity. Figure 2.10 illustrates how different functions allow one to emphasize specific parts of the scene. For instance, the function used in Figures 2.10(a) and 2.10(c) results in nearly transparent fragments on flat surfaces close the viewport, while curved surfaces are nearly opaque. The opacity quickly grows as the distance to the camera increases. This behavior allows one to place more focus on the front part of the scene. On the contrary, the function in Figure 2.10(b) places less focus in front but depicts shape on a wider range of depth. Finally, Figure 2.10(d) presents another behavior, where front parts are more opaque than distant ones. Both use the function presented in Figure 2.8(a) to modulate shading so that convexities are brightened while concavities are darkened.



(a) (b) (c)

Figure 2.11 – *(a) OIT using a k-Buffer with a constant opacity value of 0.6 (50 FPS). (b) Mean Curvature Shading only. (c) Mean curvature transparency (20 FPS). The transfer function used is on the top-right corner. Model courtesy of dessin-dus.blogspot.fr. Times are obtained with a resolution of 1000 by 1000 pixels and 16 layers.*

2.5 Results and Performance

To illustrate the benefits of extending shape depiction to transparent surfaces, Figure 2.11 shows a comparison: in Figure 2.11(a), simple OIT, in Figure 2.11(b), enhancing convexities and concavities on opaque surfaces, and finally in Figure 2.11(c) with transparency, enhanced as well with our method. OIT only allows the user to guess the internal structure of the car while Mean Curvature Shading enhances surface features only on the outer parts. On the contrary, our method enhances both internal and external parts, with a focus on the front of the scene.

To measure the impact on performance of our method, we use a synthetic test configuration, in which rendering time is only dependent of the number of objects. The scene is composed of 32 stacked full-screen quads so that all 32 layers are always filled. Each quad is located at a random depth, and is associated with a random object. We then vary the maximum number of objects, as described in Figure 2.12.

We compare the result of our solution with the brute force approach. Note that the brute-force approach has a constant rendering time, as complexity is only dependent on the number of fragments. The latter roughly corresponds to the method of Carnecky *et al.* [2013]. Implementation and renderings were done using the Open Inventor® 9.6 SDK by Thermo Fisher Scientific™ with an NVIDIA GTX 980. Results are shown on Figure 2.12 for 32 layers and a resolution of 512×512 (around 8,4 million of fragments). If our solution is obviously much slower that simple transparency (10 ms in this configuration), we can see that it improves performance for transparent shape depiction as soon as there is more than one bucket. With more than five buckets, we halve the time required, going down to a fifth of the performance of the brute force approach when there are many objects.

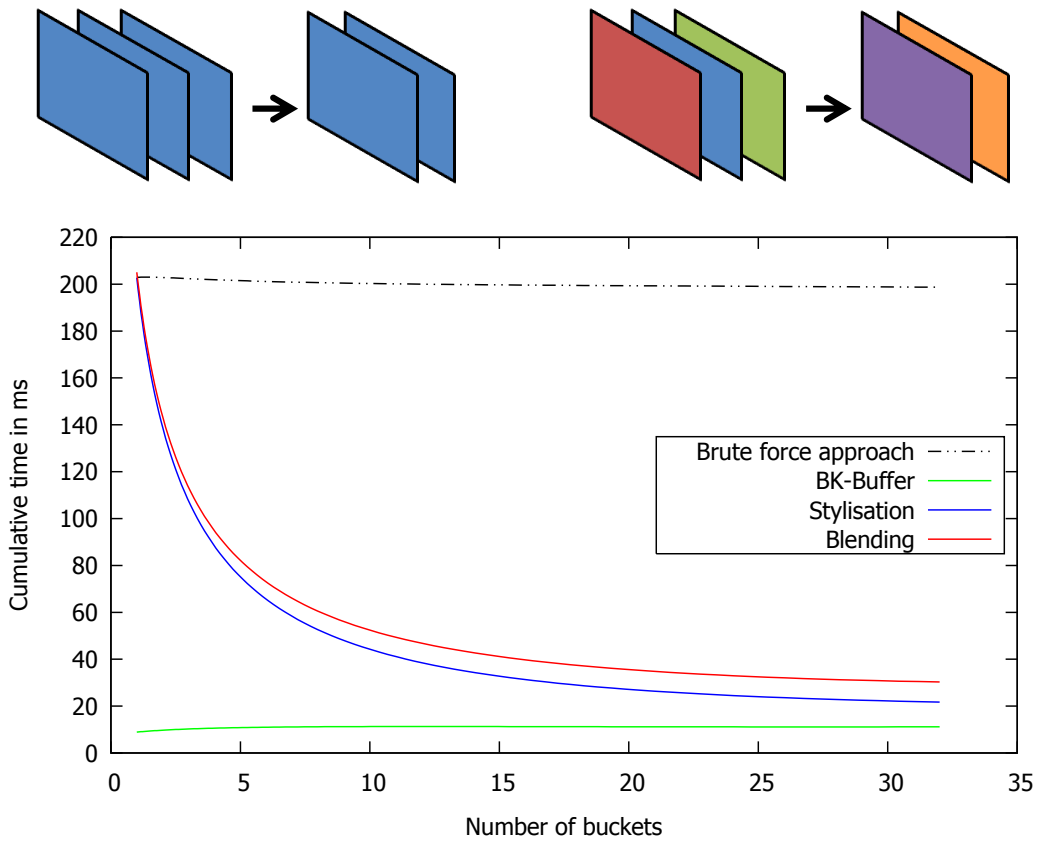


Figure 2.12 – *Cumulative rendering times of a fixed number of fragments (32) for different bucket configurations (colored quads) and 512 by 512 pixels. Full lines depict cumulative times of our method, the dotted line represents the total time of the brute force approach. As a reference, a dual-pass k-Buffer alone with constant transparency requires around 10 ms per frame.*

These observations fit rather well with theoretical gain of Section 2.3, going from $O(k^2)$ to $O(nk)$, but only when n is at least 2. Indeed, when $n \leq 2$, the cost of accessing all the fragments in a bucket becomes nearly equal to the cost of accessing the bucket, whereas for $n > 2$, the bucket cost becomes quickly insignificant with regards to the cost of accessing all fragments. However, this gain is valid only for neighbor query, as our structure still requires two sorting passes. In our algorithm, we use insertion sorting which has a complexity of $O(k^2)$ in the worst case, $O(k)$ when fragments are nearly-sorted, which is often the situation with our structure.

The last aspect concerns memory consumption. The proposed solution requires up to 128 bits per fragment (32 bits for depth and bucket ID, 64 bits for color and normal, and 32 for bucket informations) whereas a classic dual-pass k-Buffer requires 64 bits per fragment. For a full HD resolution, this over is around 32 MB per layer versus 16 MB for a simple k-Buffer.

2.6 Conclusions and limitations

We have presented a new approach as one solution to the bias in shape perception introduced by compositing multiple non-related fragments. We convey geometric properties of transparent surfaces with Mean Curvature Transparency based on the Bucketed k-Buffer. The Bk-Buffer allows an easy access to the fragments of a shape. Moreover, we provide a customizable tool to enhance scene perception for transparent surfaces. Based on a user-defined transfer function, it is possible to choose which features should be highlighted. The choice is highly dependent of which features need to be enhanced, as well as the complexity of the scene.

Our approach is dedicated to scenes with multiple objects: on a unique object, such an approach introduces too much complexity. Our structure performs better than existing ones, even if it implies an overhead in memory. However, our data structure may be optimized for more applications than neighbor query. In particular, we could use it to extend the amount of optical phenomena that we currently simulate, such as refraction, sub-surface scattering or translucency with a limited impact on performance.

On the other side, in a scene with many small objects, results can become less legible. The current stylization is based on the feedback provided by our team. A further user-study, with a representative panel, would be necessary to accurately validate the perceptual gain of our method. In future work, we also plan to investigate the impact of using line-based rendering (DeCarlo *et al.* [2003]; Ohtake *et al.* [2004]; Judd *et al.* [2007]; Kolomenkin *et al.* [2008], introduced in Section 1.4.2) to enhance third order features on transparent surfaces.

Note that such an approach is hardly generalized to pure volumes. In Chapter 3, we propose a new interactive metaphor for volume visualization and the characteristics of the required numerical scheme.

Chapter 3

Light Transport for Volume Exploration

As we detail in the first section of this chapter, the solution presented in Chapter 2 for transparent surfaces is hardly extendable to volumes. To explore such data, we introduce a new metaphor, based on medical imaging techniques (see Section 3.1.3). First done using a classical diffusion algorithm (Section 3.1.3), the solution has limited accuracy and is hardly controllable. Thus, it is not suited to be integrated directly in the Open Inventor SDK. In order to comply with this industrial constraint, we first study in Section 3.2 how we can use the RTE, as it is well suited to perform diffusion in media. We then study, in Sections 3.3 and 3.4, different numerical schemes to do so in an interactive, yet robust, way.

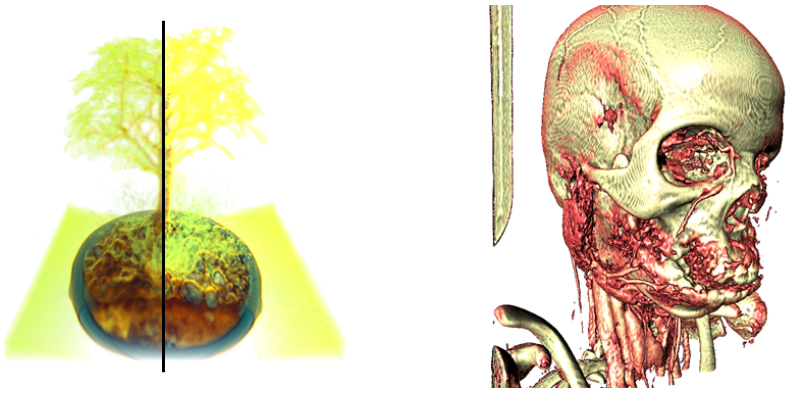
Note that for the sake of making this chapter as clear as possible, important steps and conclusions of the different sections are highlighted.

3.1 Extending to a Full Volume Data

In this section, we present how our solution for enhancing transparent surface legibility can be used when dealing with full volume data. In particular, we distinguish between isosurface rendering and ray-casting visualization. For the latter case, we also introduce another approach, inspired by fluoroscopy imaging, to improve volume visualization and exploration.

3.1.1 Volume Opacity Mapping

An obvious idea is to use the same opacity mapping than the one presented for surfaces in Chapter 2. As a reminder, we modulate the opacity of a fragment (a sample) using its local curvature and depth information. This extension to volumes can be done with two approaches: per sample (per-voxel) opacity



(a) (b)

Figure 3.1 – *Extending our previous solution for surfaces to volumes: Direct Volume Rendering (DVR) and isosurface rendering. (a) Sample-based mapping, on the left is classic DVR, on the right is DVR with our opacity mapping. Note that the mapping offers a better grasp of the configuration around rocks but can no longer see the branches, compared to classic DVR. (b) Applying our mapping to opaque approximated isosurfaces (bones in yellow, tissues in red). Note that the result amplifies noise and reconstruction limitations of the volume.*

mapping or per isosurface mapping. We now give some details about these two cases.

Per-sample Mapping When using ray-casting to generate images from a volume, the volume is evaluated per-sample. Thus, using our opacity mapping requires to compute a curvature information for each sample. This idea implies to tackle two problems:

- Is geometric curvature, as presented in Chapter 2, pertinent in a medium with a volumetric representation ?
- Our mapping handles well objects with a limited number of surfaces. Does it scale well with a large number of layers ?

Concerning the first point, a satisfying solution to compute curvature in a volume is proposed by Bruckner and Gröller [2007]: computing the curvature by comparing the normal vector from two samples along the viewing ray.

Concerning the second point, we tested our opacity mapping with a ray-casting pipeline. The result is presented in Figure 3.1(a). We can see that the mapping has a very limited impact on the legibility of the model, and, in some cases, it even has the opposite effect. Thus, for a situation like a volume in which more than a hundred samples are blended, the result of using opacity mapping does not produce a more legible visualization, despite a great overhead in computation time.

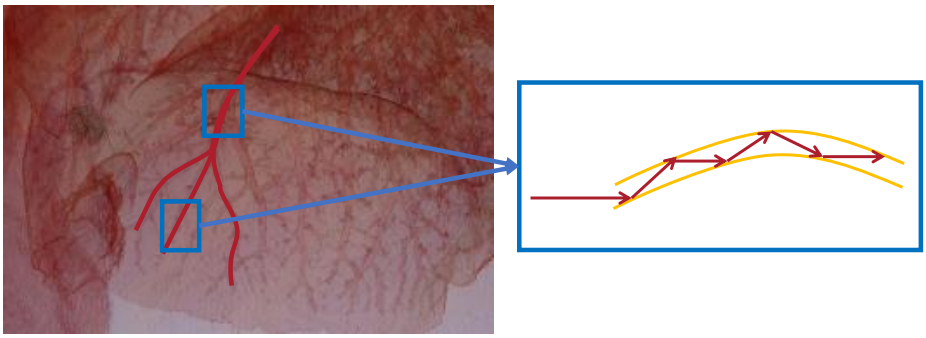


Figure 3.2 – *An illustration of a biased light transport algorithm. By using biased parameters to create or amplify discontinuities, we could enforce a confined light propagation, as illustrated on the schematic view.*

Isosurface Mapping The problematic of opacity mapping for isosurfaces is quite different from samples. In fact, it consists mostly in identifying the isosurfaces. Indeed, once that we have clearly identified surfaces inside the volume, we can directly use the method we proposed.

However, obtaining the isosurfaces is not a trivial task. The naive approach is to extract all samples with the same value, but the result will not produce legible surfaces. In this naive case, as surfaces are not well-defined, our opacity mapping can amplify noise and uncertainties more than shape cues, as presented in Figure 3.1(b). Thus, it requires to use dedicated techniques. The evaluation of these techniques is out of the scope of this thesis as we focus on light transport. For an brief overview of isosurface identitifcation algorithm, we refer the reader to Hadwiger *et al.* [2006].

Thus, assuming that we can correctly identify these surfaces, using our technique for isosurface rendering is straightforward.

The opacity mapping that we proposed in Chapter ?? does not scale well with a large number of volume samples. It is not suited for classic Direct Volume Rendering.

3.1.2 Light Propagation

Most approaches to highlight features in volumes require both data knowledge and field knowledge. In particular, segmentation-based visualization is highly task-specific and often requires to use one specific method per segmented part (e.g., Shi *et al.* [2012]). Furthermore, segmentation algorithms often operate at a global scale (e.g., Schenke *et al.* [2005]). Thus, when one wants to apply a segmentation on a restricted area, it requires many other data manipulations (e.g., cropping, transfer function adjustment, etc). Focusing on light transport,

our hypothesis is that we could use light propagation knowledge (presented in Section 1.2) to assist the visualization.

This idea is strongly inspired by fluoroscopy imaging (Mahesh [2011]). In fluoroscopy imaging, a liquid or a gaz, fluorescent when observed using X-Ray imaging, is injected in the body and tracked using a fluoroscopy microscope. This medical imaging process is used to visualize the propagation inside organs like the esophagus, the intestine or inside veins. Our hypothesis is that we can mimic this type of imaging process by adapting the volume properties to modify the output of the RTE resolution. This way, we can imagine a situation where, by adapting light transport, we can constrain, and thus guide, the light propagation in specific structures of the volume. A conceptual representation of this idea is presented in Figure 3.2.

In practice, to completely identify a part, the idea is to place a light source at a position that is known to be inside this part. Then, by adjusting the absorption and scattering parameters, the diffusion process should remain constrained into the area with similar properties.

We propose to use the RTE to reproduce the results of fluoroscopy imaging.

3.1.3 Toward a Proof of Concept: Anisotropic Diffusion

To test our hypothesis, we first focus on a pure diffusion phenomenom. We thus designed a proof of concept based on the Anisotropic Diffusion algorithm, proposed by Perona and Malik [1990]. Indeed, testing it directly using the RTE requires to evaluate the different approaches that can be used to solve the equation: path-tracing, photon-mapping, finite element resolution, iterative or not, etc. On the other hand, the Anisotropic Diffusion algorithm offers capabilities close to the one we are looking for, while being quick and easy to implement.

Anisotropic Diffusion for Image Processing

Anisotropic diffusion was initially used to perform smoothing with edge preservation. The general formulation is derived by Perona and Malik [1990] (Equation 3.1) as a general case of the heat equation that describes density changes in a material undergoing diffusion over time. They introduced a flux function C as a mean to control the diffusion process of the pixel intensity I (associated to a flux) such as:

$$\frac{\partial I}{\partial t} = C(x,y,t)\Delta I + \vec{\nabla}C^T \cdot \vec{\nabla}I \tag{3.1}$$

Note that if $C(x,y,t) = constant$, Equation 3.1 is the heat equation. Perona and Malik [1990] propose two flux functions, based on image gradients, (see

following Equations 3.2 and 3.3) that offer a trade-off between conserving the edges and smoothing homogeneous regions within these edges:

$$C(||\nabla I||) = e^{-(\frac{||\nabla I||}{\sigma})^2} \tag{3.2}$$

$$C(||\nabla I||) = \frac{1}{1 + (\frac{||\nabla I||}{\sigma})^2} \tag{3.3}$$

In these functions, the factor σ controls the sensitivity to the edges:

- If σ is high (typically $\sigma \gg 1$), the function will be close to 1, with few variations, and the diffusion will be close to the heat equation.
- If σ is low (typically $\sigma \ll 0.01$), the function will be close to 0, also with few variations and the diffusion will be nearly impossible.
- If σ is in the range $[0.01, 1]$, the function will have decent variations and the diffusion will correctly preserve edges while smoothing the homogeneous parts of the image.

Equation 3.1 can be discretized using a Forward-Time Central-Space (FTCS) method, a finite difference method often used to solve the heat equation. This method uses central differences for space derivation and forward Euler method for time derivation. Thus, Equation 3.1 becomes:

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda[C_N \cdot \delta_N I + C_S \cdot \delta_S I + C_E \cdot \delta_E I + C_W \cdot \delta_W I]_{i,j}^{t} \tag{3.4}$$

where

$$\begin{gathered} \delta_P I_{i,j} = I_P - I_{i,j} \\ C_P = C(|\delta_P I_{i,j}|) \\ N = (i-1, j) \quad S = (i+1, j) \\ E = (i, j+1) \quad W = (i, j-1) \\ 0 \leq \lambda \leq \tfrac{1}{4} \end{gathered}$$

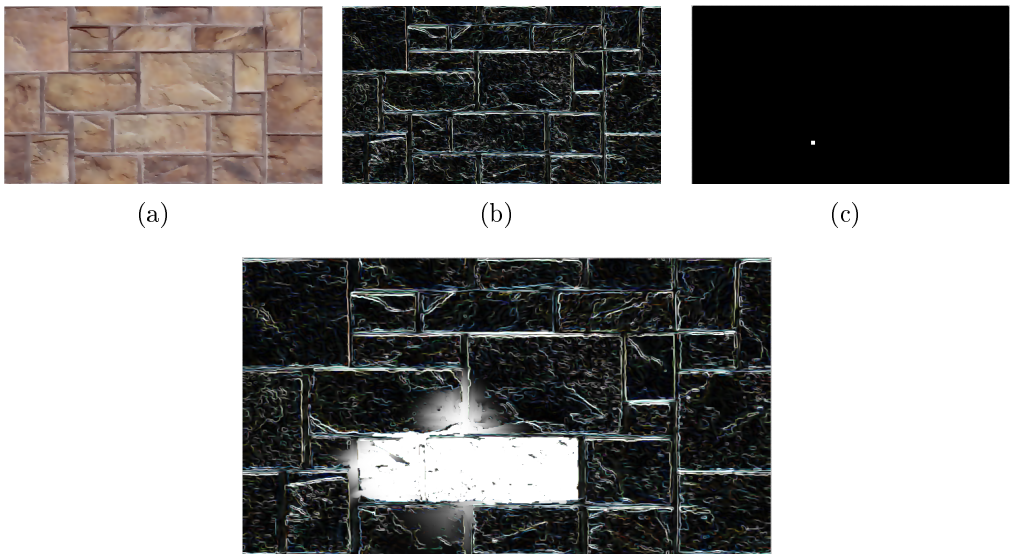
Anisotropic Diffusion for Volumes

The method can be extended to volume processing by simply taking into account the variations along the third dimension in the Equation 3.4, leading to:

$$\boxed{\begin{aligned} I_{i,j,k}^{t+1} =& I_{i,j,k}^{t} + \lambda[C_N \cdot \delta_N V + C_S \cdot \delta_S V + C_E \cdot \delta_E V + C_W \cdot \delta_W V \\ &+ C_U \cdot \delta_U V + C_D \cdot \delta_D V]_{i,j,k}^{t} \end{aligned}} \tag{3.5}$$

where

$$\begin{gathered} \delta_P I_{i,j,k} = I_P - I_{i,j,k} \\ C_P = C(|\delta_P I_{i,j,k}|) \\ N = (i-1, j, k) \quad S = (i+1, j, k) \\ E = (i, j+1, k) \quad W = (i, j-1, k) \\ U = (i, j, k+1) \quad D = (i, j, k-1) \\ 0 \leq \lambda \leq \tfrac{1}{6} \end{gathered}$$



(a) (b) (c)

(d)

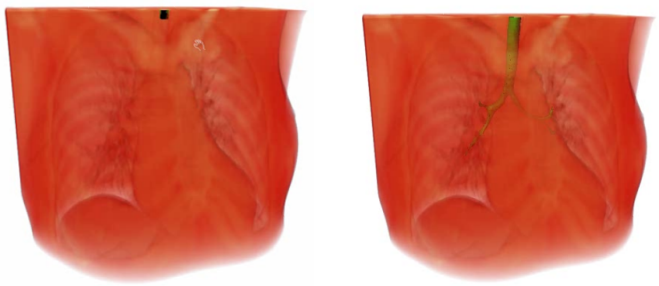
Figure 3.3 – *(a) Original image from which we want to preserve edges. (b) Gradient map of the original image, corresponding to the edges we want to preserve. (c) Energy "image", with the source and void otherwise, corresponding to image we want to smooth. (d) Energy diffusion constrained by the image gradient (visible for comprehension).*

Adapting the Algorithm for Selective Diffusion

The idea to extend the anisotropic diffusion to mimic fluoroscopy, and thus perfom selective diffusion, is depicted with a 2D example in Figure 3.3 made using Shadertoy. The idea is simple: the data that we want to "smooth" (Figure 3.3(c)) is a volume containing only void except for the energy source whereas the "edges" that we want to preserve are the one from the original data (Figure 3.3(b)). The result is a diffusion occurring mostly in smooth areas (Figure 3.3(d)).

Figure 3.4 presents the results we obtain with this algorithm. A light source has been placed at the top of the trachea (Figure 3.4(a)), and, using the gradients of the data, the diffusion (Figure 3.4(b)) is confined inside the trachea until reaching the lungs. To the best of our knowledge, the use of Anisotropic Diffusion in volumes is limited to global segmentation (e.g., Krissian [2002], Ahmed and Mohamad [2008], Morar *et al.* [2012]). Thus it has yet to be used for localized and selective diffusion as we propose to use it.

Using Anisotropic Diffusion, we have presented a solution that effectively identify specific parts of a volume. However, it presents two important limitations. The first one is that there is no absorption. It means that convergence is achieved only when each reachable voxel converges toward the same level of energy than light sources: in case of a very small overflow during the pro-



(a) (b)

Figure 3.4 – *Using the Anisotropic Diffusion to highlight the trachea in a human torso. (a) The process at time $t = 0$, the source is visible in green at the top of the trachea. (b) The result after a significant number of iterations. The area in which the diffusion occurred is depicted in green. It effectively corresponds to the trachea until it reaches the lungs.*

cess, all the volume may be flooded. The second one is that the algorithm presents limited control over the parameters, as many aspects are embedded in the model, thus it is limited in terms of rendering possibilities. The latter limitation becomes really problematic in regard to the former one, the worst case being that the whole volume will be filled.

- **Anisotropic Diffusion, adapted to volumes, can be used to identify region of interest.**
- **Due to the lack of absorption and problems of convergence, we need to limit the number of iterations to avoid overflowing.**

3.2 Solving the RTE

We have established a proof of concept to illustrate the feasibility and legibility of our idea. We now need to adopt a more practical and realistic approach by introducing absorption and a converging algorithm. Our objective is thus to reproduce this result using the RTE (Equation 1.4) that naturally integrate these two aspects. Furthermore, it must be solved in such a way that it allows one to manipulate the absorption and scattering parameters. As there are many techniques to solve this equation (some of which are briefly presented in Section 1.3.3), we need to identify the best one for our application. To do so, we first introduce the context in which the solution is developed, and then we give details about the different resolution techniques.

3.2.1 Constraints for the Resolution

In addition to the inherent difficulties of the RTE, presented in Chapter 1, in particular its strong dependency on direction, we have to comply with user-based constraints. Indeed, the solution should be integrated into Open Inventor, and thus, must provide a user-friendly interface. The most limiting part to do so is to provide an interactive solution, so that the user can modify any parameter at run-time. Indeed, interacting with a volume often requires to move frequently the camera or adjust the parameters (like the Transfer Function) to observe different parts. Finally, the solution must have a limited memory footprint. Indeed, as a feature of Open Inventor, it may be used by a wide range of users, some of whom not having high-end GPUs.

3.2.2 Resolution Methods

As introduced in Section 1.3.3, there are several kinds of algorithms to solve the RTE. We now need to confront them to our constraints to determine which one is most suited to our application.

Ray-tracing and Path-tracing techniques are widely used for the rendering quality they offer, with a small impact on memory. However, the resulting image is generally computed according to a given point of view, thus, it must be recomputed whenever the camera is moved or when parameters are modified. Even if it is possible with recent implementations to generate an image in a matter of seconds, like Cinematic Rendering (Comaniciu *et al.* [2016] and Engel [2016]), the frequent modifications performed while observing the volume greatly hinder the interactivity of the methods. The same goes for Photon-Mapping, but with a non-negligible additional memory cost as a photon map must be stored for the full volume.

The other kind of approaches is to use implicit representations: expressing the problem as a linear system to obtain the radiance value at each point. This system is then solved either directly (e.g., LU factorization, Cholesky factorization, etc.) or solved iteratively. The former requires to store the whole matrix of the system and operate on it, which quickly becomes really heavy in terms of both memory consumption and computation time. Thus, it suffers from the same limitation as the previous methods when the parameters are changed. The latter is less subject to this problem as the iterative pattern allows one to compute each iteration with a local kernel, on a per-voxel basis. This iterative pattern allows the user to modify the parameters between the iteration and the system should stabilize itself if it is well defined.

From all the above considerations, we can assume that using an implicit representation with an iterative resolution should be a good candidate for our application. Furthermore, our prototype with Anisotropic Diffusion gave us the hint that seeing the diffusion process until convergence, instead of directly

seeing the converged result, could help in understanding the process and correct it if necessary.

- **Visualizing the diffusion process, step by step, helps in understanding the volume structure.**
- **An iterative resolution with implicit representation is best suited to reach this goal.**

3.3 Iterative Resolution using Finite-Elements

After identifying iterative resolution as the most suited method for our application, we present our first try at using it. As a reminder, the RTE (Equation 1.4) is presented below:

$$\begin{aligned}\vec{\omega}^T \cdot \vec{\nabla}_p L(p,\vec{\omega}) = &- K_t(p) \cdot L(p,\vec{\omega}) + Q_e(p,\vec{\omega}) \\ &+ \int_{4\pi} K_s(p) \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,\vec{\omega}') d\vec{\omega}'\end{aligned}$$

The first step is to find an iteration-based approach of Equation 1.4. We also need to handle the directional terms (depending on $\vec{\omega}$) of this equation which are the most troublesome terms if we pre-calculate any term.

3.3.1 Ignoring In-Scattering

As a first step, we focus only on absorption and out-scattering:

$$\vec{\omega}^T \cdot \vec{\nabla}_p L(p,\vec{\omega}) = Q_e(p) - K_t(p) L(p,\vec{\omega}) \tag{3.6}$$

Furthermore, we consider that the energy source (Q_e), placed inside the volume at position p_0, is isotropic and constant:

$$Q_e(p,\vec{\omega}) = Q_e(p) = \begin{cases} Q_0 & \text{if } p = p_0 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Equation 3.6, applied to the whole volume data, can be expressed as a linear system:

$$ML = Q_e - RL$$

Thus, solving Equation 3.6 for the whole volume is equivalent to solving this linear system. To do so, we use an iterative process by decomposing R into two separate matrices:

$$M + R = A + B$$

where A would be at best diagonal, at worst triangular. This allows us to reformulate:

$$AL^t = Q_e - BL^{t-1} \tag{3.8}$$

We must now obtain the matrices M and R (A and B). We compute them by using a finite element method for which Equation 3.6 is a strong form of a partial derivatives equation so the first step is to obtain its weak form.

3.3.2 Representation of the Quantities

We now need to establish the weak form of the equation. To do so, we must first introduce the test-functions ψ and φ that we use to solve the system and approximate our solution according to:

$$L(p, \vec{\omega}) = \sum_j \sum_l L_{j,l} \psi_j(\vec{\omega}) \varphi_l(p) \tag{3.9}$$

For the spatial function, let $(\varphi_l(p))$ be a basis of piece-wise linear function, with $l \in [0, N]$, N being the total number of voxels:

$$\varphi_l(p) = \begin{cases} 1 + p & \text{if } p_{l-1} \leq p \leq p_l \\ 1 - p & \text{if } p_l \leq p \leq p_{l+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Concerning the directional function, we simply use a function $(\psi_j(\vec{\omega}))$ that projects onto the different principal axes, with $j \in [0, 3]$:

$$\begin{aligned} \psi_0(\vec{\omega}) &= 1 \\ \psi_1(\vec{\omega}) &= \vec{\omega}.x \\ \psi_2(\vec{\omega}) &= \vec{\omega}.y \\ \psi_3(\vec{\omega}) &= \vec{\omega}.z \end{aligned}$$

Compared to other possibilities like Spherical Harmonics or Radial Functions for example, this representation has the advantage of being straightforward and should not require too many operations for further implementation. Furthermore, this choice was strongly dictated by its similarity to an approximation commonly used in light diffusion problems: the Diffusion Approximation.

Introduced in Computer Graphics by Stam [1995] (and later used by Jensen *et al.* [2001]), the Diffusion Approximation is widely used to address scattering problems. As stated by Stam [1995] (and Pierrat [2007] with a physicist point of view), in highly dispersive media (low absorption and high scattering) there are many occurrences of scattering events (multiple scattering). Thus,

even if the phase function is highly anisotropic, if the media is wide and dense enough, the radiance tends to become isotropic due to these repeated deflections. The same idea is widely used with microfacet-based BRDFs which are often represented with a Lambertian behavior in case of a rough surface. We can then reformulate the radiance in the RTE as a combination of an isotropic contribution, $\phi(p)$, and a direction-dependent correction term, $\vec{\omega}^T \cdot \vec{E}(p)$, such as:

$$\boxed{L(p,\vec{\omega}) = \frac{1}{4\pi}\phi(p) + \frac{3}{4\pi}\vec{\omega}^T \cdot \vec{E}(p)} \tag{3.11}$$

This approximation can also be obtained with a first order expansion in the Legendre polynomial basis (the P_1 approximation, P_n refering to the Legendre polynomial for n order). More details can be found in Pierrat [2007].

The isotropic term is linked to the spherical integral of the outgoing radiance, called the fluence rate:

$$\phi(p) = \int_{\Omega^2} L(p,\vec{\omega})d\omega \tag{3.12}$$

The direction-dependent term corresponds to the total flux crossing a surface, called the irradiance vector, and is defined by:

$$\vec{E}(p) = \int_{\Omega^2} L(p,\vec{\omega})\vec{\omega}d\omega \tag{3.13}$$

The factors $\frac{1}{4\pi}$ and $\frac{3}{4\pi}$ are normalization constants.

3.3.3 Finite Element Approximation

The weak form of Equation 3.6 with bases $\psi_i(\vec{\omega})$ and $\varphi_k(p)$ is

$$\begin{aligned}\int_{\Omega^2,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\vec{\omega}^T \cdot \nabla_p L(p,\vec{\omega}) = \\ \int_{\Omega^2,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)Q_e(p) \\ - \int_{\Omega^2,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)L(p,\vec{\omega})K_t(p)\end{aligned} \tag{3.14}$$

By using the projection of the radiance function in the same bases (see Equation 3.9), we get:

$$\begin{aligned}\sum_j \sum_l L_{j,l} \int_{\Omega^2,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\vec{\omega}^T \cdot \nabla_p(\psi_j(\vec{\omega})\varphi_l(p)) = \\ \int_{\Omega^2,[-1,1]} \psi_j(\vec{\omega})\varphi_l(p)Q_e(p) \\ - \sum_j \sum_l L_{j,l} \int_{\Omega^2,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\psi_j(\vec{\omega})\varphi_l(p)K_t(p)\end{aligned} \tag{3.15}$$

To go further, we assume that our data is locally homogeneous, meaning that for a given position and its neighborhood, $K_t(p) = K_t$. Furthermore, we assume that the light source is isotropic ($Q_e(p, \vec{\omega}) = Q_e(p)$) and punctual, and thus expressing Q_e in our system is straightforward: we just have to modulate the original function by 3π (details are presented in Appendix A.1).

To obtain the final formulation of our system (Equation 3.8), we must first obtain the matrices M and R from Equation 3.15. To do so, we can decompose our matrices coefficient $M_{i,j,k,l}$ and $R_{i,j,k,l}$ to evaluate separately spatial and directional terms.

Thus, we have:

$$M_{i,j,k,l} = \vec{\alpha}_{i,j}^T \cdot \vec{\beta}_{k,l} = \int_{4\pi} \psi_i(\vec{\omega})\psi_j(\vec{\omega})\vec{\omega}^T \cdot \int_{[-1,1]} \varphi_k(p)\nabla_p\varphi_l(p) \tag{3.16}$$

The mathematical details to obtain the coefficients $M_{i,j,k,l}$ are presented in Appendix A.1 and gives us for $\vec{\alpha}_{(i,j)}$:

$$\vec{\alpha}_{(i,j)\in[0,3]^2} = \begin{bmatrix} (0,0,0) & (\frac{4\pi}{3},0,0) & (0,\frac{4\pi}{3},0) & (0,0,\frac{4\pi}{3}) \\ (\frac{4\pi}{3},0,0) & (0,0,0) & (0,0,0) & (0,0,0) \\ (0,\frac{4\pi}{3},0) & (0,0,0) & (0,0,0) & (0,0,0) \\ (0,0,\frac{4\pi}{3}) & (0,0,0) & (0,0,0) & (0,0,0) \end{bmatrix} \tag{3.17}$$

For $\vec{\beta}_{k,l}$, we use a formulation more adapted to a 3D volume (of size $W \times H \times D$): $(k,l) \in [0,N]^2 \iff ((k_x,k_y,k_z),(l_x,l_y,l_z)) \in ([0,W],[0,H],[0,D])^2$. That way, we can express the system as a reduced kernel that is computed for a voxel at position k. With the details from Appendix A.1, we can also reduce (k,l) to $k+d, d \in \{-1,0,1\}^3$. All other combination of (k,l) results in zeros in the matrices, thus our spatial kernel is a $3 \times 3 \times 3$. Furthermore, we established in Appendix A.1 that:

$$\vec{\beta}_{k,l} = (\beta_{k,l}, \beta_{k,l}, \beta_{k,l})$$

Thus, we only need to express the formulation for $\beta_{k,l}$, that corresponds to a tensor β_d, centered on the position k, such that:

$$\beta_{d\in\{-1,0,1\}^3} = \frac{1}{9}\left[\begin{bmatrix} \frac{-1}{8} & 0 & \frac{1}{8} \\ \frac{-1}{2} & 0 & \frac{1}{2} \\ \frac{-1}{8} & 0 & \frac{1}{8} \end{bmatrix} \begin{bmatrix} \frac{-1}{2} & 0 & \frac{1}{2} \\ -2 & 0 & 2 \\ \frac{-1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{-1}{8} & 0 & \frac{1}{8} \\ \frac{-1}{2} & 0 & \frac{1}{2} \\ \frac{-1}{8} & 0 & \frac{1}{8} \end{bmatrix}\right] \tag{3.18}$$

Then, by using the same logic for $R_{i,j,k,l}$, we have:

$$R_{i,j,k,l} = K_t\gamma_{i,j}\delta_{k,l} = \rho \int_{4\pi} \psi_i(\vec{\omega})\psi_j(\vec{\omega}) \int_{[-1,1]} \varphi_k(p)\varphi_l(p) \tag{3.19}$$

The mathematical details to obtain the coefficients $R_{i,j,k,l}$ are presented in Appendix A.2 and gives us first for $\vec{\gamma}_{(i,j)}$:

$$\gamma_{(i,j)\in[0,3]^2} = \begin{bmatrix} 4\pi & 0 & 0 & 0 \\ 0 & \frac{4\pi}{3} & 0 & 0 \\ 0 & 0 & \frac{4\pi}{3} & 0 \\ 0 & 0 & 0 & \frac{4\pi}{3} \end{bmatrix} \tag{3.20}$$

For $\delta_{k,l}$, we also use a 3D notation ($(k,l) \iff k+d$), and we also obtain a tensor δ_d, centered on the position k, such that:

$$\delta_{d\in\{-1,0,1\}^3} = \left[\begin{bmatrix} \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \end{bmatrix} \begin{bmatrix} \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{2}{27} & \frac{8}{27} & \frac{2}{27} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \end{bmatrix} \begin{bmatrix} \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \end{bmatrix} \right] \tag{3.21}$$

We can note that with this formulation, we have a factor 64 between the current element ($d = (0,0,0)$) and its furthest neighbors ($d = (\pm 1, \pm 1, \pm 1)$). Thus, our assumption that our data is locally homogeneous should not introduce an important bias.

From the above information, we can construct the diagonal matrix A from the coefficient $\gamma_{i=j}$ and $\delta_{d=(0,0,0)}$, while the matrix B is composed of the remaining terms. Thus, the final formulation to evaluate the radiance per voxel is presented in Equation 3.22, in which only the non-zero elements from γ and α are kept. This formulation is given using the notations and normalization from the Diffusion Approximation (Equation 3.11).

$$\begin{bmatrix} \phi_j \\ E_j.x \\ E_j.y \\ E_j.z \end{bmatrix} = \frac{81}{32\pi}\frac{1}{K_t} \begin{bmatrix} \phi_e \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ -\frac{27}{24K_t}\sum_d \beta_d \begin{bmatrix} 3(E_{j+d}.x + E_{j+d}.y + E_{j+d}.z) \\ \phi_{j+d} \\ \phi_{j+d} \\ \phi_{j+d} \end{bmatrix} - \frac{27}{8}\sum_d \delta_d \begin{bmatrix} \phi_{j+d} \\ E_{j+d}.x \\ E_{j+d}.y \\ E_{j+d}.z \end{bmatrix} \tag{3.22}$$

3.3.4 Limitations

We have formulated the finite element system. We must now check if it is compatible with a Jacobi iterative resolution. Our system can be reformulated with the form $Ax = b$ where x is the radiance. Such a system is ensured to converge with the Jacobi method only if A is a diagonally dominant matrix, otherwise, the convergence is not guaranteed. As a reminder, a matrix A with coefficients a_{ij} is diagonally dominant if:

$$|a_{ii}| \geq \sum_{j\neq i} |a_{ij}|$$

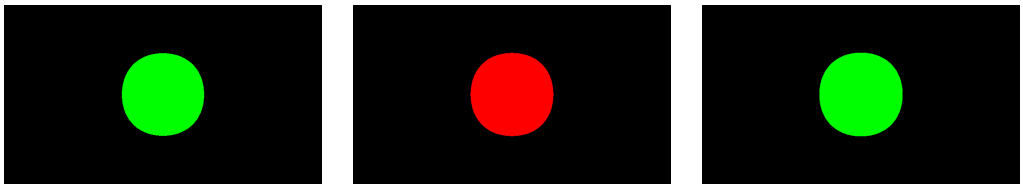


Figure 3.5 – *Alternation between positive (in green) and negative (in red) values over three iterations. The images are from a 2D implementation on Shadertoy.*

In our case, whatever the value that is used for K_t, this condition is not verified and thus we have no guarantee of the convergence of the method. This problem was verified with a 2D version (using Shadertoy). The result was effectively unstable as the values rapidly grew far beyond the value of Q_e while oscillating between positive and negative values (see Figure 3.5). Furthermore, for this version, we did not consider multiple scattering, which would amplify this instability by increasing even further the rightmost member of the inequality condition.

If our matrix is not diagonally dominant, it is a symmetric positive-definite matrix. A Gauss-Seidel pattern, which converges in this case, may solve the system. However, providing efficient implementations on GPUs of such a solver is still an open problem in Computer Science. Moreover, in our case, this solver would require to frequently reorganize our data, between computation and rendering steps, which could result in a significant impact in term of computation time.

Furthermore, the system presented has a intrinsic flaw: the extinction parameter is only present as an inverse form, which limits the type of media that we can address (e.g., $K_t \approx 0$ is not possible).

- **Whatever the value that is used for K_t, the stability condition of the finite element system is not verified and thus we have no guarantee of the convergence of the method.**
- **The steady RTE in the Diffuse Approximation is not compatible with a Jacobi iterative resolution.**

3.4 The Unsteady RTE

Due to the limitations that we presented above, we chose to focus on the unsteady version of the RTE (Equation 1.4). Indeed, this version, compared to the steady one, has the advantage of complying well with an iterative resolution. In this section, we first discuss the case of applying directly a finite difference scheme to the unsteady RTE. Then, due the limitations of this approach, we present how it can be expressed using the directional moments of

the radiance, with a finite difference scheme also.

3.4.1 Using Time Finite Differences

As a reminder, the equation we now use is:

$$\frac{1}{c}\frac{\partial L(p,t,\vec{\omega})}{\partial t} + \vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega}) = -K_t(p) \cdot L(p,t,\vec{\omega}) + Q_e(p,t,\vec{\omega}) + \int_{\Omega^2} K_s(p) \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i(p,t,\vec{\omega}')d\vec{\omega}'$$

We keep the assumption that the source is isotropic, and we also consider that it does vary in time: $Q_e(p,t,\vec{\omega}) = Q_e(p)$.

By using a forward finite difference scheme to discretize the time derivation, we naturally obtain an iterative-compliant formulation:

$$L^{t+1}(p,\vec{\omega}) = L^t(p,\vec{\omega}) + c \cdot dt \cdot \left(\vec{\omega}^T \cdot \vec{\nabla}_p L^t(p,\vec{\omega}) - K_t(p) \cdot L^t(p,\vec{\omega}) + Q_e(p) + \int_{\Omega^2} K_s(p) \cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}') \cdot L_i^t(p,\vec{\omega}')d\vec{\omega}' \right)$$

This method is very close to the Discrete Ordinate Method (DOM) introduced by Chandrasekhar [1950]. We use a finite set of directions to solve the equation. The slight difference is that DOM uses an angular discretization only to evaluate the scattering term, the directional derivative being solved along the x, y and z axes.

Limitations

This formulation does not present the same problem as the finite element solution we presented. However, it is highly sensible to the chosen angular distribution for both the directional derivative and the scattering integral. Indeed, if there is not enough directions, these directions become clearly visible in the final image. This is the same limitation as for DOM, as stated by Jönsson *et al.* [2013]. Thus, to obtain a legible result, we need to use a large number of directions, but this hinders greatly the computation cost and thus, the interactivity of the method.

Using a finite differences scheme directly on the unsteady RTE produces artifacts along the principal directions.

3.4.2 Using a Moment-based Formulation

We have briefly introduced and explained the Diffusion Approximation (Equation 3.11) for the finite element method. What is interesting with this approximation is the link with the moments of radiance. Thus, by using these same moments, applied directly to the whole RTE, we can obtain a system that does not depend on radiance but on its moments.

Before presenting how we express the RTE in this context, we introduce a notation for the moment operator, thus from now on, we note μ_n the moment of order n:

$$\mu_n(f(\vec{\omega})) = \int_{\Omega^2} f(\vec{\omega})\vec{\omega}^n d\omega$$

Order 0 (μ_0)

$L(p,t,\vec{\omega})$: For the order 0, two terms are straightforward: $L(p,t,\vec{\omega})$ and $\partial L(p,t,\vec{\omega})/\partial t$. By definition, the order 0 of these two terms is the fluence rate:

$$\mu_0(L(p,t,\vec{\omega})) = \phi(p,t)$$
$$\mu_0(\frac{\partial L(p,t,\vec{\omega})}{\partial t}) = \frac{\partial \phi(p,t)}{\partial t}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega})$: For this term, we have (details in Appendix B):

$$\mu_0(\vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega})) = \vec{\nabla}_p{}^T \vec{E}(p,t)$$

$\mathcal{P}(\vec{\omega},\vec{\omega_i})L_i(p,t,\vec{\omega_i})$: The simplification of this term depends on how we consider L_i. The details of the simplification are presented in Appendix B, and leads to:

$$\mu_0\left(\int_{4\pi} \mathcal{P}(\vec{\omega},\vec{\omega_i})L_i(p,t,\vec{\omega_i})d\omega_i\right) = \phi(p,t)$$

Order 1 (μ_1)

$L(p,t,\vec{\omega})$: For the order 1, $L(p,\vec{\omega},t)$ and $\frac{\partial L(p,t,\vec{\omega})}{\partial t}$ are also straightforward. By definition, the order 1 of these two terms is the irradiance vector :

$$\mu_1(L(p,t,\vec{\omega})) = \vec{E}(p,t)$$
$$\mu_1(\frac{\partial L(p,t,\vec{\omega})}{\partial t}) = \frac{\partial \vec{E}(p,t)}{\partial t}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega})$: Once again, the mathematical details are provided in Appendix B and we have:

$$\mu_1(\vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega})) = \frac{1}{3}\vec{\nabla}_p \phi(p,t)$$

$\mathcal{P}(\vec{\omega}, \vec{\omega_i})L_i(p,t,\vec{\omega_i})$: Finally, for this term (Appendix B) we have the following formulation:

$$\mu_1 \left(\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) L_i(p,t,\vec{\omega_i}) d\omega_i \right) = g\vec{E}(p,t)$$

where g is the anisotropic factor.

Unsteady RTE with Diffuse Approximation

By combining all the results presented above, we can finally obtain our linear system, described by Equations 3.23 and 3.24, with $K_{t'} = K_a + (1-g)K_s$.

$$\boxed{\begin{aligned} \frac{1}{c}\frac{\partial \phi(p,t)}{\partial t} &= -div(\vec{E}(p,t)) - K_a(p)\phi(p,t) + \phi_e(p,t) \quad (3.23) \\ \frac{1}{c}\frac{\partial \vec{E}(p,t)}{\partial t} &= -\frac{1}{3}\vec{\nabla}_p \phi(p,t) - K_{t'}(p)\vec{E}(p,t) + \vec{E}_e(p,t) \quad (3.24) \end{aligned}}$$

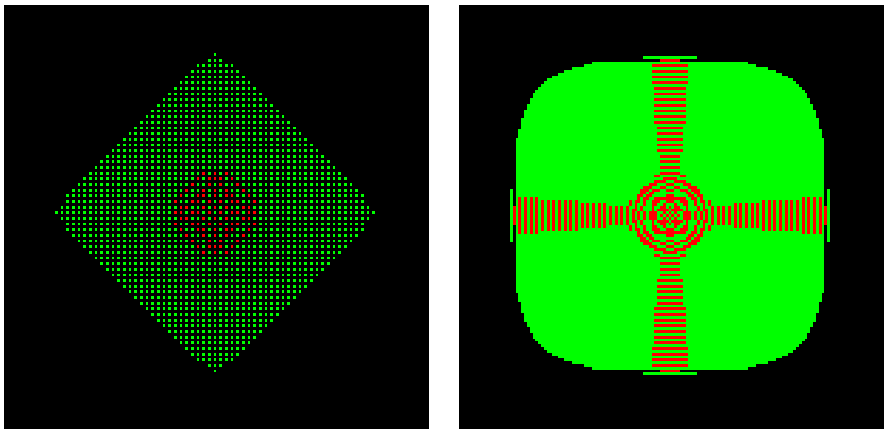
This system was implemented using a finite differences pattern (FTCS):

$$\begin{aligned} \frac{\partial f(p,t)}{\partial t} &\iff \frac{f^{t+1}_{i,j,k} - f^t_{i,j,k}}{\Delta t} \\ \frac{\partial f(p,t)}{\partial x} &\iff \frac{f^t_{i+1,j,k} - f^t_{i-1,j,k}}{2\Delta x} = \frac{\delta_x f^t}{2\Delta x} \\ \frac{\partial f(p,t)}{\partial y} &\iff \frac{f^t_{i,j+1,k} - f^t_{i,j+1,k}}{2\Delta y} = \frac{\delta_y f^t}{2\Delta y} \\ \frac{\partial f(p,t)}{\partial z} &\iff \frac{f^t_{i,j,k+1} - f^t_{i,j,k+1}}{2\Delta z} = \frac{\delta_z f^t}{2\Delta z} \end{aligned}$$

With this discretization, Equations 3.23 and 3.24 lead to Equation 3.26. For this formulation, we consider that $\Delta x = \Delta y = \Delta z = \Delta$ such that:

$$\boxed{\begin{aligned} \phi^t_{i,j,k} =& c\Delta t \phi_{e,(i,j,k)} + (1 - K_{t,(i,j,k)} c\Delta t)\phi^{t-1}_{i,j,k} \\ & - c\frac{\Delta t}{2\Delta}(\delta_x E.x^{t-1} + \delta_y E.y^{t-1} + \delta_z E.z^{t-1}) \quad (3.25) \\ \vec{E}^t_{i,j,k} =& c\Delta t \vec{E}_{e,(i,j,k)} + (1 - K_{t,(i,j,k)} c\Delta t)\vec{E}^{t-1}_{i,j,k} - c\frac{\Delta t}{6\Delta} \begin{pmatrix} \delta_x \phi^{t-1} \\ \delta_y \phi^{t-1} \\ \delta_z \phi^{t-1} \end{pmatrix} \quad (3.26) \end{aligned}}$$

The system described by Equation 3.26 should converge toward a solution if $(K_{t,(i,j,k)} c\Delta t, c\frac{\Delta t}{2\Delta})) < (1,1)$. This means that we must take a time step Δt very small compared to Δ and have $K_t \leq \frac{1}{2\Delta}$. As an example, if we take the values for a human liver, we have $K_t \approx 25cm^{-1} = 2500m^{-1}$ (L Sandell and Zhu [2011]), we should have $\Delta \leq 5.10^{-4}$. Thus to ensure convergence, we must have $\Delta t \leq 10^{-11}s$.



(a) (b)

Figure 3.6 – *Using the moment-based methods leads to a grid pattern of positive (in green) and negative (in red) values. The images are from a 2D implementation on Shadertoy. (a) Based on a restricted neighborhood (4 direct neighbors). (b) Based on a extended neighborhood (4 direct and 4 diagonal neighbors). In (b), the problem is attenuated but still present.*

However, this implementation has a flaw that we did not anticipate. The pattern of Equation 3.26 leads to an alternation between the propagation of ϕ and $\vec{E}$, as one can not significantly change as long as the other has not changed as well. The derivation pattern combined with this effect produces significant artifacts, presented in Figure 3.6 with a 2D implementation (using Shadertoy). If using directly the six neighbors ($+/-$ for $x/y/z$) for the derivation, the result will be a grid with an alternation of positive, negative and null values (Figure 3.6(a), in 2D). By considering a weighted neighborhood, this problem is reduced but artifacts remain in the principal directions (Figure 3.6(b), in 2D).

Using a moment-based formulation up to order 1 leads to grid artifacts with an alternation between positive and negative values.

Extending to Order 2 (μ_2)

To solve the aforementioned issue, we consider using an extended radiance representation. It consists in adding the second order moment in the approximation. Thus, we go from:

$$L(p,\vec{\omega}) \approx C_0\mu_0(L(p,\vec{\omega}) + C_1\vec{\omega}^T \cdot \mu_1(L(p,\vec{\omega})) = \frac{1}{4\pi}\phi(p) + \frac{3}{4\pi}\vec{\omega}^T \cdot \vec{E}(p)$$

To the new one:

$$L(p,\vec{\omega}) \approx C_0\mu_0(L(p,\vec{\omega}) + C_1\vec{\omega}^T \cdot \mu_1(L(p,\vec{\omega})) + C_2\vec{\omega}^T\mu_2(L(p,\vec{\omega}))\vec{\omega}$$
$$= C_0\phi(p) + C_1\vec{\omega}^T \cdot \vec{E}(p) + C_2\vec{\omega}^T S\vec{\omega}$$

in which S is a symmetric tensor that can be interpreted as the radiative pressure. Furthermore, the moment of order 0 can be integrated into the diagonal of the tensor for clarity. In practice, this must be done to keep a consistent directional basis. Thus, after regrouping and evaluating the normalization factor, we consider:

$$L(p,\vec{\omega}) \approx \frac{3}{4\pi}\vec{\omega}^T \cdot \vec{E}(p) + \frac{5}{4\pi}\vec{\omega}^T S\vec{\omega}$$

With this representation, we need a set of 9 equations to obtain all necessary values: 3 for $\vec{E}$ and 6 for S (as S is symmetric). After development, reorganization and simplification (available in Appendix B, Equations B.11 and B.12), we obtain a first set of 12 equations (before applying the symmetry) presented in Equation 3.27 (simplified by $\frac{4\pi}{3}$ and $\frac{4\pi}{5}$ respectively).

$$\frac{1}{c}\frac{\partial\vec{E}(p,t)}{\partial t} + \frac{1}{5}\begin{pmatrix}\delta_x tr(S) + 2(\delta_x S_{xx} + \delta_y S_{xy} + \delta_z S_{xz})\\ \delta_y tr(S) + 2(\delta_y S_{yy} + \delta_x S_{xy} + \delta_z S_{yz})\\ \delta_z tr(S) + 2(\delta_z S_{zz} + \delta_x S_{xz} + \delta_y S_{yz})\end{pmatrix} = \vec{E}_e(p,t) - K_t(p)\vec{E}(p,t)$$

$$\frac{1}{c}\frac{\partial}{\partial t}(2S(p) + tr(S(p))I) + \begin{bmatrix}2\delta_x E_x + div(\vec{E}) & (\delta_x E_y + \delta_y E_x) & (\delta_x E_z + \delta_z E_x)\\ (\delta_x E_y + \delta_y E_x) & 2\delta_y E_y + div(\vec{E}) & (\delta_y E_z + \delta_z E_y)\\ (\delta_x E_z + \delta_z E_x) & (\delta_y E_z + \delta_z E_y) & 2\delta_z E_z + div(\vec{E})\end{bmatrix} =$$
$$(2S_e(p) + tr(S_e(p))I) - K_t(p)(2S(p) + tr(S(p))I) \tag{3.27}$$

However, once we obtain this, we can see that the process that produced artifacts is still present as we still have this alternation effect between the pure directional component and the more isotropic one.

Extending up to order 2 moments does not solve the grid problems presented in the solution with up to order 1 moments.

3.5 Conclusion

We have presented a study about several numerical techniques to solve the RTE. From this study, we reach the same statement that the one Koerner *et al.* [2018] have recently presented on the difficulties of solving the RTE with implicit representations. This equation is hardly solved by using standard

approaches like finite element or finite differences methods without leading to significant artifacts.

Concerning our specific application, we can say that the steady RTE is not compatible with a Jacobi iterative resolution. It can however be solved iteratively with a Gauss-Seidel pattern combined with an adapted data organization. However, the latter is hardly compatible with our initial goal of interactivity. Furthermore, the method only ensures convergence but not necessarily a plausible result at each step. Indeed, we could have temporary negative radiance values at a random step while still obtaining the correct values after convergence.

For a more natural iterative resolution, the unsteady RTE is more suited as, in theory, each step give radiance results that are consistent with this quantity, even before convergence. For this reason, and despite the issues presented in Section 3.4, this version remains the best suited to our needs. Thus, we explore in Chapter 4 a new formulation based on an additional hypothesis to the Diffuse Approximation.

Chapter 4

Radiance Diffusion Equation

In the previous chapter, we introduced a new metaphor to explore volumes and we presented some considerations about numerical schemes. It emerged first that the steady RTE was hardly compatible with iterative Jacobi resolution, and second, that the unsteady version needs further considerations to be used. Thus, in this chapter, we first focus on the diffusion model that we use to solve the unsteady RTE. Then we present the different tools that we currently use to interact with this algorithm, to manipulate the parameters and the lighting configuration. Next, we discuss about the first results obtained on medical data and the performance and limitations of the method. Finally, we propose some ideas to enhance the interaction with the light sources, that have yet to be fully tested.

4.1 Toward a Diffusion Model

We have presented in Chapter 3 the main issues that we faced in solving iteratively the RTE. We now introduce the solution that we use to solve the RTE in such a way that it allows us to manipulate the parameters as much as possible. Our solution relies on an approximation, used in many light diffusion problems, that we first present, along with the resulting system. Then we discuss about the intrinsic limitations and the validity of the chosen approach. The results and implementation details are presented later in Section 4.2.

4.1.1 Establishing the Diffusion Equation

To obtain an equation that has the form of a diffusion equation, we have to make asumption, in addition to the Diffusion Approximation. As it is done in some diffusion problems (e.g., Haskell *et al.* [1994]), we consider that the irradiance vector is approximately steady, meaning that:

$$\frac{\partial \vec{E}}{\partial t} \approx 0$$

Note that this hypothesis can also be inferred directly from the Diffuse Approximation. As the latter is valid only for long time scales, it implies that the flux (the irradiance vector $\vec{E}$) can not present significant variations, to ensure that we are in a diffusive state. This is also inherent from the formulation of Equation 3.11, as, to ensure that the radiance is positive, we must have $\phi \gg ||\vec{E}||$ (Ishimaru [1997]).

We now use this consideration to further simplify the moment-based formulation of the unsteady RTE we have presented in Chapter 3. As a reminder, we had for the unsteady version (Equations 3.23 and 3.24):

$$\frac{1}{c}\frac{\partial \phi(p,t)}{\partial t} = -div(\vec{E}(p,t)) - K_a(p)\phi(p,t) + \phi_e(p,t)$$
$$\frac{1}{c}\frac{\partial \vec{E}(p,t)}{\partial t} = -\frac{1}{3}\vec{\nabla}_p\phi(p,t) - K'_t(p)\vec{E}(p,t) + \vec{E}_e(p,t)$$

By assuming that the source is purely isotropic, we may omit its directional component ($\vec{E}_e(p,t)$). Equation 3.24 is then reduced to:

$$\boxed{\vec{E}(p,t) = -\frac{\vec{\nabla}_p\phi(p,t)}{3K'_t(p)}} \tag{4.1}$$

Note that Equation 4.1 is analogous to Fick's law (Fick [1855]) which is widely used in diffusion problems. As stated by Haskell *et al.* [1994], this approximation is valid with the unsteady RTE when we focus on biological tissues. As this type of media is the main target of our application, we can reasonably consider that we are in scope of this approximation.

With this hypothesis, we can greatly simplify the system presented in Section 3.4.2. Also, we use the more adapted diffusion coefficient D (in meters **m**), defined as:

$$\boxed{D(p) = \frac{1}{3K'_t(p)}} \tag{4.2}$$

The expression of Equation 4.1 is then simplified to:

$$\vec{E}(p,t) = -D(p)\vec{\nabla}_p\phi(p,t)$$

Note that this expression can be found in Computer Graphics in Jensen *et al.* [2001] but in their case, it was derived using the Diffuse Approximation with the steady RTE (Equation 1.5), as mentioned earlier.

We can now replace the irradiance vector in Equation 3.23 to obtain an equation depending only on ϕ:

$$\frac{1}{c}\frac{\partial \phi(p,t)}{\partial t} = div(D(p)\vec{\nabla}_p\phi(p,t)) - K_a(p)\phi(p,t) + \phi_e(p,t)$$

Finally, after developing the divergence operator, and if we do not consider K'_t to be homogeneous, which is the case for medical data, we obtain:

$$\boxed{\frac{1}{c}\frac{\partial\phi(p,t)}{\partial t} = D(p)\Delta_p\phi(p,t) - \vec{\nabla}_p D(p)^T \cdot \vec{\nabla}_p\phi(p,t) - K_a(p)\phi(p,t) + \phi_e(p,t)} \tag{4.3}$$

If we have a homogeneous medium, $\vec{\nabla}_p D(p) = \vec{0}$, we obtain the diffusion equation often used in Physics to simulate infinite homogeneous media, as presented by Pierrat [2007].

4.1.2 The Finite Difference System

We use a FTCS scheme to discretize Equation 4.3 which gives us Equation 4.4. Note that we once again use {N,S,W,E,U,D} to identify the position: North (N) is $(i{+}1,j,k)$, South (S) is $(i{-}1,j,k)$, etc. The absence of index corresponds to the central position. Also, for the sake of clarity, we incorporate the factor $c\Delta t$ directly into ϕ_e.

$$\begin{aligned}\phi^t &= \phi_e + (1 - c\Delta t K_a)\phi^{t-1} \\ &+ D\frac{c\Delta t}{\Delta^2}(\phi_N + \phi_S + \phi_W + \phi_E + \phi_U + \phi_D - 6\phi)^{t-1} \\ &- \frac{c\Delta t}{4\Delta^2}\begin{vmatrix} D_N - D_S \\ D_W - D_E \\ D_U - D_D \end{vmatrix} \cdot \begin{vmatrix} \phi_N - \phi_S \\ \phi_W - \phi_E \\ \phi_U - \phi_D \end{vmatrix}^{t-1}\end{aligned} \tag{4.4}$$

However, this formulation is not practical as it involves four distinct variable parameters that can influence the stability condition. These are Δt, Δ, K_a and D. Theoretically, they all can vary between 0 and $+\infty$. Before studying the range that these parameters can vary within to ensure convergence (which is presented in the next section), we reformulate Equation 4.4 using terms that are easier to manipulate. Thus, we use the notion of scattering albedo α (van de Hulst [1981]), which quantifies the ratio between scattering and absorption in a medium, such as:

$$\alpha = \frac{K_s}{K_t} \tag{4.5}$$

The advantage of this parameter is that it varies between 0 and 1:

- $\alpha \approx 0$: $K_a \gg K_s$, we have an dominantly absorbing medium.
- $\alpha \approx 1$: $K_s \ll K_a$, we have a dominantly scattering medium.

We can use this parameter to find a direct relation between K_a and the coefficient D presented in Equation 4.2. To do so, we first reformulate to express K_s:

$$K_s = \frac{\alpha}{1-\alpha}K_a$$

We then replace K_s in the expression of D with a term depending on α to finally obtain a direct relation with D:

$$\boxed{K_a = \frac{1-\alpha}{3(1-\alpha g)D}} \tag{4.6}$$

Note that this relation implies that D can not be exactly 0 which is consistent with the definition of D (Equation 4.2). It can however be close to 0, thus, whenever we talk about D varying from 0 to something, it is a simplification meaning a value close to zero. Using this relation, we have replaced a parameter varying between 0 and $+\infty$ with one varying between 0 and 1.

Finally, we also introduce a new notation for the diffusion coefficient using a unitless parameter β such as:

$$\boxed{D = \beta\Delta} \tag{4.7}$$

Thus, we can reduce to a formulation where we no longer have explicitly the terms $\frac{c\Delta t}{\Delta^2}$ and $c\Delta$ but only the ratio $\frac{c\Delta t}{\Delta}$. The final formulation is the following:

$$\begin{aligned}\phi^t = \phi_e + \left(1 - \frac{c\Delta t}{\Delta}\frac{1-\alpha}{3(1-\alpha g)\beta} - 6\beta\frac{c\Delta t}{\Delta}\right)\phi^{t-1} \\ +\beta\frac{c\Delta t}{\Delta}(\phi_N + \phi_S + \phi_W + \phi_E + \phi_U + \phi_D)^{t-1} \\ -\frac{c\Delta t}{4\Delta}\begin{vmatrix}\beta_N - \beta_S \\ \beta_W - \beta_E \\ \beta_U - \beta_D\end{vmatrix} \cdot \begin{vmatrix}\phi_N - \phi_S \\ \phi_W - \phi_E \\ \phi_U - \phi_D\end{vmatrix}^{t-1}\end{aligned} \tag{4.8}$$

In Equation 4.8, the controllable parameters are: D the diffusion coefficient, indirectly through β, K_a, indirectly through β and α, Δt the time step and Δ the spatial step. Concerning g, we will see in Section 4.1.3 that it will be fixed to $g = 0$.

4.1.3 Considerations about the Model

Link to Anisotropic Diffusion

The first notable aspect of this model (Equation 4.3) is that it has a significant similarity with the Anisotropic Diffusion formulation. As a reminder, Anisotropic Diffusion (in 2D) is:

$$\frac{\partial I}{\partial t} = C(x,y,t)\Delta I + \vec{\nabla}C^T \cdot \vec{\nabla}I$$

We know, from Section 3.1.3, that this equation is stable using a Finite Difference scheme with an iterative pattern. Thus we can reasonably assume that, due to the similarity, our system should be stable as well. This means there should be a set of parameters for which this model satisfies the Jacobi convergence condition.

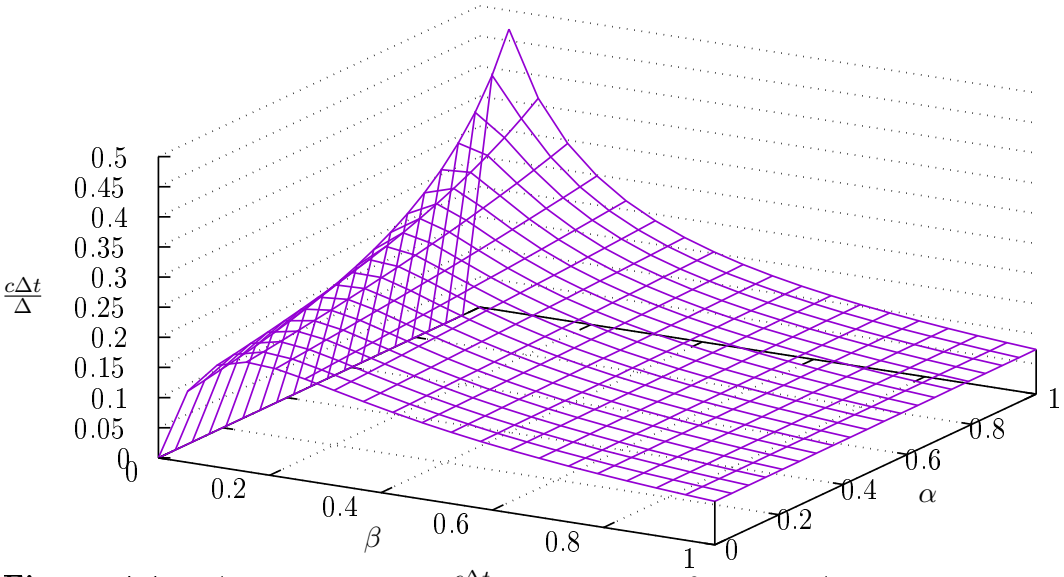


Figure 4.1 – *A plot of the ratio $\frac{c\Delta t}{\Delta}$ depending on β and α. All values below the curves satisfy the stability condition for its corresponding β and α.*

Adding User-Based Conditions

Before studying the stability conditions for our system, we introduce some constraints in order to ensure that the application is user-friendly. In particular, this tool may be employed by users not well-versed in light transport. Thus, we decided to expose only a limited control over the parameters such that:

- Values not ranging from 0 to 1 should not be directly exposed.
- Diffusion should be either controlled directly with a value ranging from 0 to 1 or indirectly using predefined mapping depending on the desired effect. This imply that $\boxed{\beta \in [0;1]}$.
- Absorption should be controlled either automatically or by using the albedo α.

Furthermore, in order to simplify the conditions and reduce the number of degrees of freedom that could lead to divergence, we first fix $\boxed{g = 0}$. This means that we consider the scattering to be isotropic (uniform phase function). Doing so makes it impossible to address many media in a realistic way but greatly reduces the complexity of the model. Note that this consideration is also comforted by our experiments, in which the impact of this factor was very limited compared to the other parameters.

Stability Conditions

We now need to determine the range of the parameters for which the system is stable. As a reminder, we have a system with the form $Ax = b$ where x is

the radiance. A is diagonally dominant if:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$$

In our case:

$$|a_{ii}| = \left| 1 - \frac{c\Delta t}{\Delta} \frac{1-\alpha}{3\beta} - 6\beta \frac{c\Delta t}{\Delta} \right|$$

This term can not be negative, as it would mean that an element can loose more energy than it possesses, which is impossible. Thus we can eliminate the absolute value operator. As for the right member, we have:

$$\sum_{j \neq i} |a_{ij}| = 6\beta \frac{c\Delta t}{\Delta} + \frac{c\Delta t}{2\Delta} \Big(|\beta_N - \beta_S| + |\beta_W - \beta_E| + |\beta_U - \beta_D| \Big)$$

In the worst case, we have maximum gradient on the diffusion coefficient: β goes from 1 to 0, meaning that $\max(|\beta_N - \beta_S|) = \max(|\beta_W - \beta_E|) = \max(|\beta_U - \beta_D|) = 1$. This allows us to obtain an inequality that links $\Delta t/\Delta$ to β and α:

$$1 - \frac{1-\alpha}{3\beta} \frac{c\Delta t}{\Delta} - 6\beta \frac{c\Delta t}{\Delta} \geq 6\beta \frac{c\Delta t}{\Delta} + \frac{3}{2} \frac{c\Delta t}{\Delta}$$

This finally gives us the following condition that has to be respected when determining Δt and Δ, depending on β and α:

$$\boxed{\frac{c\Delta t}{\Delta} \leq \frac{6\beta}{72\beta^2 + 9\beta + 2(1-\alpha)}} \tag{4.9}$$

The rightmost part of Equation 4.9 is plotted in 3D, in Figure 4.1, to explicit the range within which the ratio can vary, depending on the parameters. Thus, any value below the curve ensures that we are in the stability domain. In particular, we can see that for a ratio of $\frac{c\Delta t}{\Delta} \leq 0.1$, we are under the curve for nearly all values of β and α can be used. This is encouraging as it means that by fixing, $\frac{c\Delta t}{\Delta} = 0.1$, we can address most of the possible configurations. As an example using this value, for a data with $\Delta = 1mm$, we have $\Delta t \approx 3.33 \times 10^{-13} s$. In practice, it may be interesting to locally adjust Δt, as long as the condition is verified, to allow a faster diffusion. This latter process should, however, not be used when one is targeting accuracy more than depiction, as it is clearly not realistic.

4.2 Application for Enhanced Visualization

We have introduced the model we adopt as well as its domain of validity and we now explain how we use it. First, we give details about how we implement

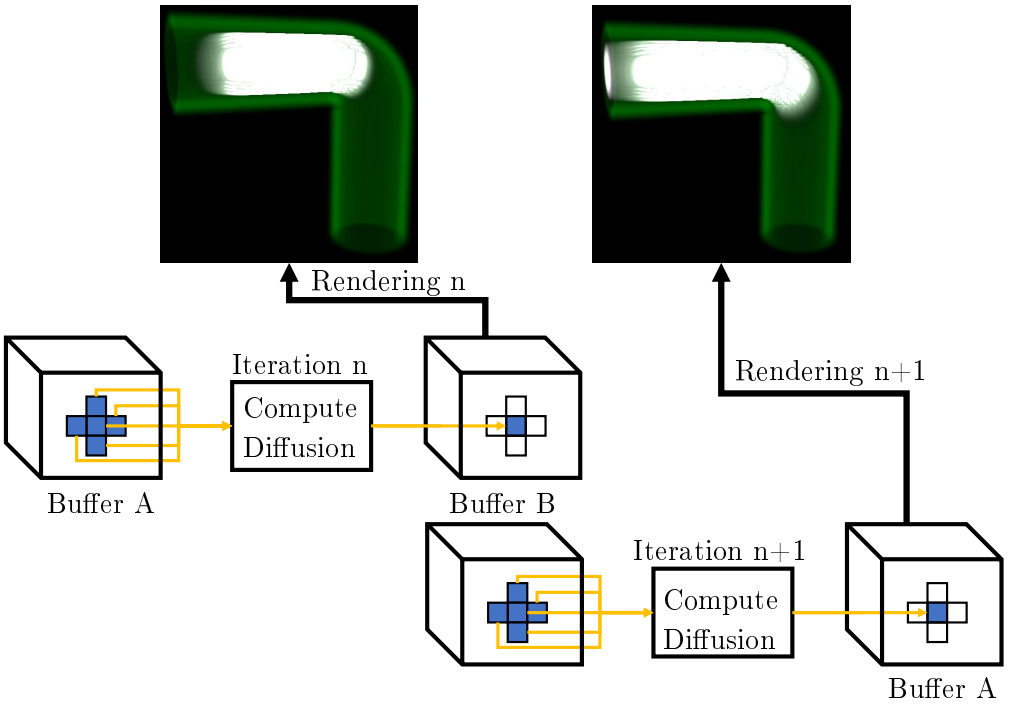


Figure 4.2 – *To solve the diffusion process, we use a Ping Pong scheme. The input of the compute shaders alternates between Buffer A and Buffer B at each iteration. For rendering, the output buffer is then used during ray-cast, thus, if the frame n uses Buffer B for rendering, the frame n+1 will use Buffer A.*

it, how we control the parameters and the interaction tools that we use, then we present how this solution effectively addresses our problem.

Note that the work presented in this section is the current state and is still in development. Thus, some hypotheses and considerations we use, to design the tools for user interaction, have not been formally validated. They are mostly based on our experience and the knowledge of developers and application engineers from the Open Inventor team as well as their feedback.

4.2.1 Implementation details

Our implementation is done using Open Inventor (version 9.8) and OpenGL (up to version 4.3) to handle features not directly supported by Open Inventor. We first present how we implement the diffusion algorithm and then, how we store the results.

Our implementation relies on two 3D buffers to store the result of the diffusion, using a Ping Pong scheme (Gray and Reuter [1992]). This is necessary as we use an iterative resolution, meaning that during the computation we access data that comes from a previous step. Thus, we must ensure that the value we get has not been overwritten by the result of another computation. As

illustrated in Figure 4.2, the first buffer (A) is used as source for one iteration (n) and the result is stored in the second one (B). Then B is used as input for both the current rendering (n) and the computation for the next iteration ($n+1$). The result is then stored into buffer A, which is then used for rendering ($n+1$), and so on.

As for the computation step, it is done with compute shaders. As evoked in Section 1.3.1, a compute shader is a shader stage that can be used without invoking the graphics pipeline (see Brown *et al.* [2012] for the OpenGL specification). This stage allows us to exploit the parallelism of a GPU for other purposes than rendering, while using a buffer format that can be directly interfaced with the programmable stages of the graphics pipeline.

As compute shaders are not present in Open Inventor, it was added to the SDK to implement this part of the work. However, the current implementation is synchronized with the graph traversal. This means that the computing is synchronized with the frame rate.

4.2.2 Controlling the Parameters

Based on the formulation we have chosen, the controllable parameters in our algorithm are: β, using either predefined functions or using a custom transfer function, and K_a, indirectly using either an automatic value based on β or by using the albedo α. The results obtained by using some of these options are presented later in Section 4.2.4. In the next paragraphs, we briefly explicit the different options that we propose.

Setting β We propose three mapping options. For each option, it is also possible to use the complementary $(1-\beta)$, depending on the use case.

- **Custom transfer function**: this option maps a scalar value (from the data) to a value for β. It offers a total control to the user, as long as the values are in the range $[0;1]$. As such, the containment capabilities of this option are limited.
- **Scalar gradient:** this option uses the gradient of the data as input for a function close to the flux function introduced in Section 3.1.3 (Equation 3.2):

  $$\beta(||\nabla S||) = e^{-(\frac{||\nabla I||}{\sigma})^2}$$

  σ controls the importance given to the gradient (see Section 3.1.3). This option performs well for constrained diffusion as it relies on local data variations.
- **Gradient of the transfer function:** this option allows one to combine both a user-defined mapping with a transfer function and containment capabilities offered by using gradients.

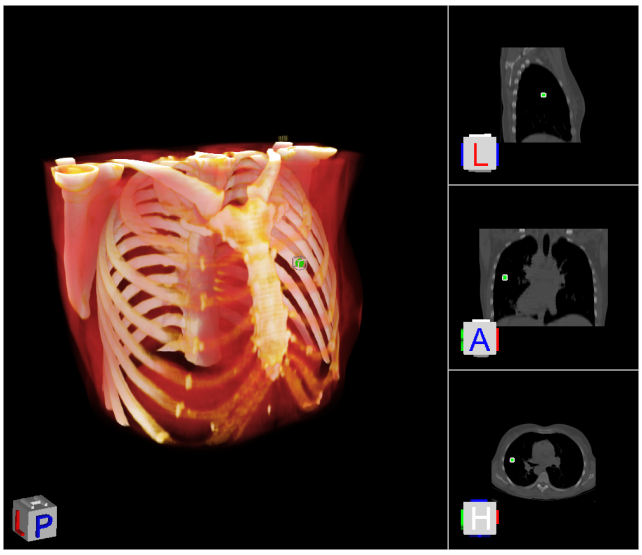


Figure 4.3 – *The set-up used to place the light source: three orthoslices corresponding to the three main axes are used to move the source along each one. The light source is represented, on both slices and the ray-casted view, by a green sphere.*

Setting K_a We offer two possibilities to set this parameter:

- **Taking the inverse of β:** Despite being non-physical, this mapping is useful for selective diffusion. Indeed, it ensures that absorption is low in area with maximum diffusion, and high when diffusion is minimum. Also, with this mapping, the possibilities of being outside the stability domain are very limited.
- **Using the albedo α:** This mapping provides a more physically accurate solution for most values. However, to ensure that the computation does not diverge, the albedo may be clamped to stay within the convergence range (presented in Figure 4.1).

4.2.3 Controlling the Sources

The manipulation of the light sources is an important aspect of the application. Indeed, if they are not placed correctly, the result would be, at best, completely inconsistent with the expected result. In this case, at least the user knows the source is ill-placed. But at worst, it could be sufficiently close to the expected result, and implies that it is the correct one, while still introducing a significant bias. In the following paragraphs, we describe how we place light sources in a 3D space.

In our implementation, we can place up to eight light sources. The main difficulties we encountered while manipulating these sources are their placement in a 3D space. Indeed, to correctly place a source, we can either set the position by specifying its (x, y, z) directly, use draggers to move the source in the scene, pick a position with the mouse...

Our experiments and feedbacks from Application Engineers on this matter led us to several conclusions:

- Placing the source directly is very tedious, in particular when sources are not immediately placed at the correct spot. If they need to be moved, this solution is not practical.
- Using draggers is also tedious as it requires to constantly rotate the object to effectively interact in 3D.
- Picking a position directly in 3D is ambiguous, as we try to select a point in 3D by using a screen without liable feedback on the depth.

Thus, following these conclusions, we chose to use a combination of orthoslices to place the sources, as illustrated in Figure 4.3. We use one orthoslice per reference axis (X/Y/Z), combined with a slider to navigate along each axis. Using slices allows the user to see exactly where is the source in the volume and what is around it without being disturbed by occluding information.

Note that this solution is still limited as it is done using only sliders, which are not optimal for this purpose. This aspect is partially addressed in Section 4.2.5.

4.2.4 Results

In this section, we present the results we achieved with our model and the different features and tools we have introduced. First, we present performance of the solution. We then discuss the possible achievements of our methods based on some examples of renderings using different sets of parameters.

Performance

The measures presented in this section are made on a laptop workstation with an i7-6820HQ CPU, 32 GB of RAM and an NVIDIA Quadro M5000M GPU (8 GB of V-RAM). We first present some time references, then we discuss about the memory usage of the method, and finally, we discuss its current limitations.

Time To evaluate the computation time, we measure the time per frame on a synthetic test case (Figure 4.2). We do this for four resolutions: 128^3, 256^3, 512^3, 1024^3. The averaged results are presented in Table 4.1, along with the total time required per frame without diffusion as a reference. All results are in milliseconds (**ms**).

We can see that for small volumes, the algorithm has a very limited cost. In this case, it should be very beneficial to decorrelate the computation and the frame display. As a reference, for a computing volume of $128 \times 128 \times 128$, the method of Zhang and Ma [2013] requires around 52ms for one light source, and up to 91ms for five light sources, to solve directly the RTE. However, for

Size	Total time per frame	Reference time	Diffusion overhead
128^3	8.9	6.4	2.5
256^3	28.9	13.3	15.6
512^3	133.9	26.7	107.2
1024^3	759.3	49.9	709.4

Table 4.1 – *Time required per frame with and without computing as well the difference between the two. Results are presented on a synthetic volume with four different resolutions. All results are in milliseconds (**ms**).*

large volumes, the computing time is significant and may be a burden when manipulating the data.

Also, the time per voxel decreases as the number of voxels increases. The probable cause lies in the way the dispatch command, for the compute shaders, is executed. In our implementation, it separates the volume in a fixed number of groups, regardless of its size. Groups too small (or too large) are not well adapted to compute shaders (Brown *et al.* [2012]). Indeed, the cost of dispatching the computation over a group of only a few elements is significant compared to the speed gain of processing this group in dedicated threads.

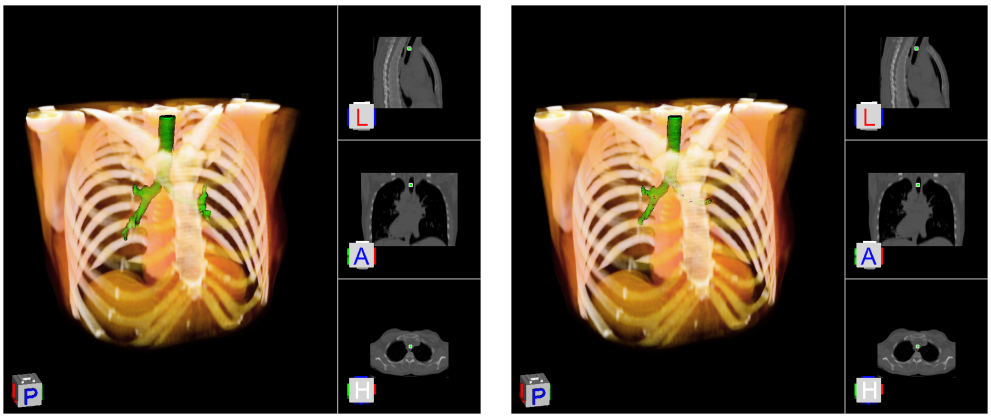
Finally, it is important to note that the total rendering time (computing and ray-casting) is highly dependent on the chosen visualization, as ray-casting can be costly if the data is mostly transparent.

Also, due to the graph traversal of Open Inventor and the interactions with the state, it is difficult to correctly measure the effective computation time.

Memory Footprint In term of memory cost, our solution is probably not optimal. Indeed, in the current implementation, we theoretically either need two additional volumes with one floating point per original voxel or one with two floating point channels. This implies a huge memory overhead that we need to address in the future. In practice, as memory was not an issue on the test station for most of the test volumes, the current solution uses two textures with four floating point channels each. This allows a speed up in the computation as we can store data-dependent gradients during the first iteration to avoid recomputing them. This is already accounted for in the computing cost presented above, thus, eliminating this additional memory can not be done until further optimization.

Finally, in the current state, we also have to make a copy of the original data. This copy is due to the way Open Inventor handles volume data. Indeed, the data is loaded as a texture on the GPU only when the ray-casting starts, thus we can not access the original volume from outside of the VolumeViz rendering pipeline. This implies a memory overhead as well as a delay when the application starts to do the copy, or whenever the original data is modified.

To sum up the memory overhead, assume we have a volume of dimensions



(a) (b)

Figure 4.4 – *Comparison between our method (b) and our prior prototype with Anisotropic Diffusion (a) . Diffused energy is depicted in green.*

$N = W \times H \times D$, N being the total number of voxels. Our application currently needs $256 \times N$ bits in addition to the original volume and its copy. For examples, for a volume of $128 \times 128 \times 128$ voxels with 8 bits per voxel, the applications needs:

- 2 MB for the volume.
- 2 MB for its copy.
- 64 MB for the diffusion buffers.

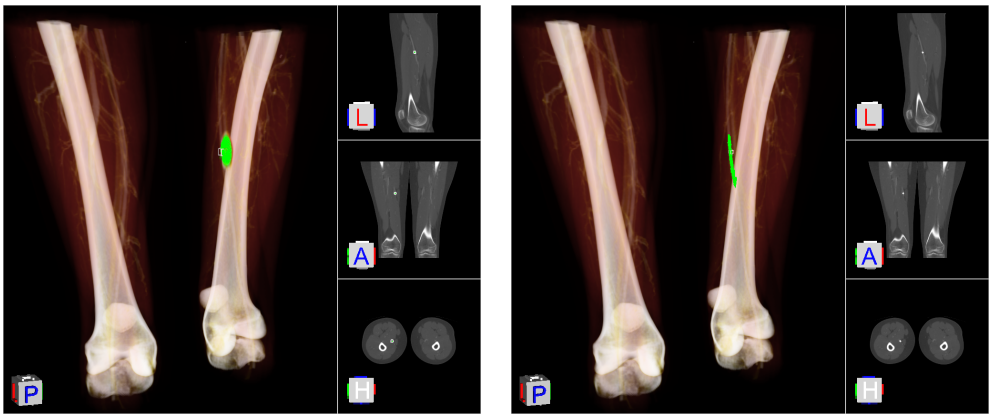
We can see that for large volumes, the memory overhead will quickly exceed the memory limits even of modern hardwares.

Results

We present, in Figures 4.4(a) and 4.4(b), a comparison of our method with our original prototype using Anisotropic Diffusion. We can see that, using the same visualization, our method achieves results a bit more constrained than with our first prototype. This is consistent with the fact that the RTE includes an absorption term. The parameters used here are: scalar gradient for β and $1 - \beta$ for K_a.

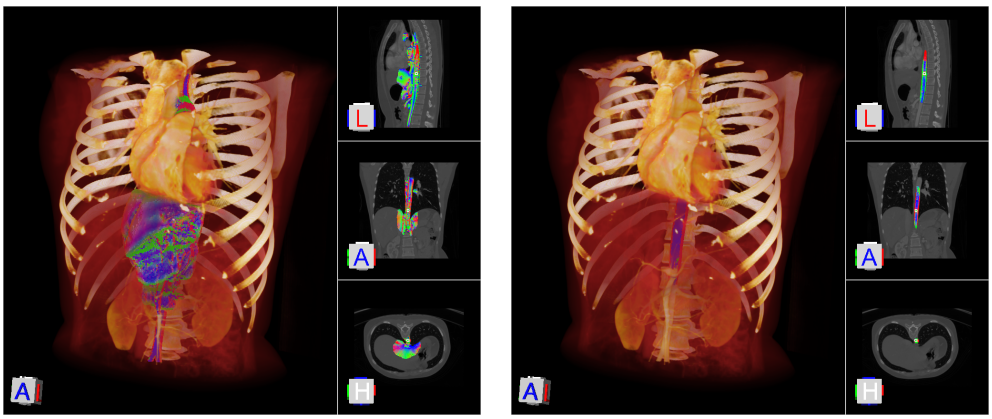
Our experiments proved that using a gradient-based mapping offers the best containment capabilities. This is illustrated in Figures 4.5(a) and 4.5(b). In Figure 4.5(a), β is mapped to the opacity given by the transfer function whereas in Figure 4.5(b), β is mapped to its gradient. The latter performs better that the former in offering a constrained diffusion into the vein where the source is located.

The work presented here concerns mostly the underlying algorithm, thus, it is the starting point for more investigations to overcome its limitations. In particular, we have yet to find the visualization solution that is best suited to



(a) (b)

Figure 4.5 – *Different results obtained with our method (diffused energy is depicted in green). (a,b) We compare two mapping for β: (a) using the opacity value from the transfer function, (b) using its gradient.*



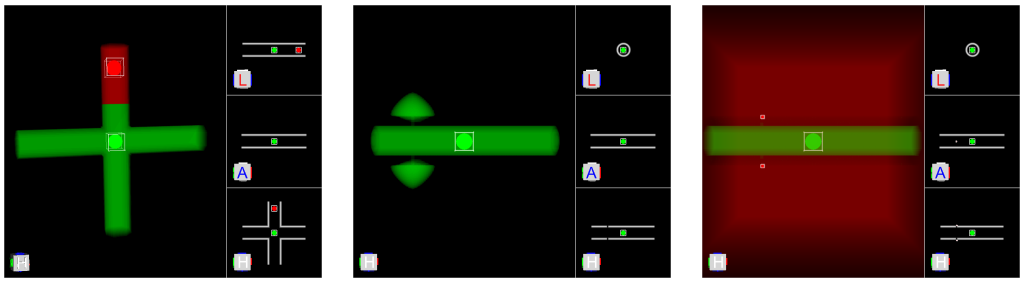
(a) (b)

Figure 4.6 – *Examples of ill-controlled parameters by visualizing the normalized irradiance vector ($\vec{E}$). The vector is also displayed on the slices for better understanding. (a) The diffusion is done without enough absorption, leading to overflowing. (b) The diffusion is done with too much absorption, leading to an over-constrained diffusion, as the vein is not fully identified.*

our application.

Also, a limitation that we have already talked about is overflowing, illustrated in Figure 4.6(a). Even if it is possible to adjust the parameter to attenuate the problem of overflowing, it often results in over-constrained diffusion (Figure 4.6(b)). Furthermore, it is not possible to completely get rid of the risk of overflowing, as there are often artifacts and uncertainties in the original data.

Finally, even with the tools we have presented, the interaction is limited, in particular for the design of the lighting configuration. Indeed, placing the



(a) (b) (c)

Figure 4.7 – *Examples of using negative sources with a synthetic case, with positive values in green, negative in red. (a) Example of using a negative source to block the diffusion. (b,c) In a tube with two holes, the diffusion overflows (b) which can be limited using negative sources (c) to define an area that should not block the diffusion.*

sources using only sliders allows the user to place correctly a source but it remains a tedious process.

In the next sections, we propose ideas using biased light sources directly in the diffusion as a starting point to address some limitations. We also introduce new tools and considerations to help the user in the task of choosing the correct lighting configuration.

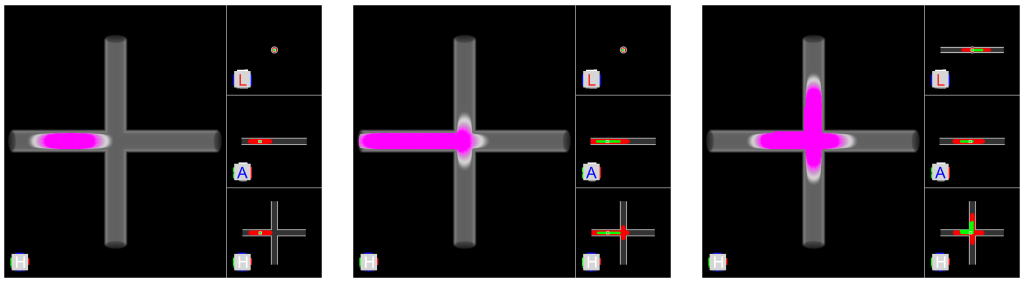
4.2.5 Using the Sources as a Tool

In this section, we present ideas that have been tested on a synthetic case: a pipe placed inside a homogeneous medium, and filled with another homogeneous medium similar to the first one. These proof of concepts are tools to: first, reduce overflowing using negative light sources, and second, manipulate more efficiently the light sources, their type, shape, and position.

Negative Sources One of the limitations of the algorithm is overflowing. If the data contains uncertainties and acquisition artifacts, the diffusion will not remain constrained in a structure.

Our idea to address this limitation is to introduce negative light sources in the process. We then use the diffusion to propagate negative energy, as presented in Figure 4.7. When the negative energy meets with the positive one, its creates a virtual barrier where the positive and negative energies nullify each other (Figure 4.7(a)). These sources can then be used to prevent some overflows by "patching" points where leaks occur, as illustrated in Figures 4.7(b) and 4.7(c).

Even though the idea is completely not physical, as energy can not be negative, it still behaves as predicted, as the mathematical model we use does not embed any consideration about the sign of the quantities. In addition to limiting the problem of overflowing, using negative light sources also gives the



(a) (b) (c)

Figure 4.8 – *Examples of painting with the light sources, with a synthetic case. The light source is indicated in green on the slice, the rest of the diffused energy is depicted in red in the slice, in purple in the volume. (a) The initial light source, consistent with the sphere manipulator. (b) Using the previous diffusion in (a), the light source is extended to a tube. (c) An example of curved light source, to initiate the diffusion in two of the four tube parts.*

possibility to define areas in which we do not want the diffusion to occur, which can also be used to prevent any leak (Figure 4.7(c)).

Painting with Light Another option to manipulate the sources is the possibility to make it persistent. This way, the user can use the light source like a paint brush to define a custom light source. This tool can be useful in several cases:

- When the user has already a high degree of confidence in an area of the data.
- When the user has gained a high degree of confidence from a previous diffusion.
- When the user wants to use complex light sources.

The main advantage of this tool is to offer the possibility to design a complex lighting environment. This feature is however tedious with the current implementation as the source is moved using sliders. To offer a more natural interaction, picking actions should be added.

Forcing the Diffusion We have just introduced the tool we implemented to limit the diffusion and offer customizable light sources, we now present the one we use to force the diffusion. Indeed, in some cases the user may want to speed up the process, in particular when there is a high degree of confidence in the area already reached.

We introduced above a solution to specify extended sources. However, it requires user interaction which may not be desired. Thus, we also propose a solution to offer the possibility to directly convert an area in which energy has been diffused into an extended light source. This allows the user to re-inject energy and thus speed up the diffusion process. The main drawback of this

solution is that the converged result will not be the same as with the original light sources. However, as the purpose of this solution is not to provide a physically accurate result but to provide the user a simple speedup, this issue should not be addressed until proven otherwise.

Finally, in case of a flaw not detected until the re-injection, it could quickly lead to overflowing.

4.3 Conclusion and Future Work

Achievements

Our goal was to provide a solution to visualize structures and regions of interest in volumes without using global segmentation. To address this problem, we have proposed a model to solve the RTE iteratively, with its stability condition. This method fulfills the initial goal as it effectively provides a fluid-like diffusion while being based on light transport.

We have shown in Section 4.2.4 that the solution can be used to identify regions of interest using diffusion. We also introduced a variety of tools to interact with the diffusion process, to offer as much control as possible while avoiding a divergence in the solution.

However, the method remains limited as most of the presented features and their limitations have yet to be addressed and validated. In particular, the visualization aspect has not been properly studied in terms of rendering styles and transfer functions. Also, the method could probably be optimized to reduce the memory footprint as well as the computation cost.

Future Work

The most important remaining work is to study which quantities should be displayed (e.g., the fluence rate, the irradiance vector, the radiance...) and how it should be displayed (e.g., directly the intensity, using a dedicated transfer function...). As we first focused on the diffusion model, these aspects are not treated here and should be addressed in future work.

Another aspect of the current state is the limitations in terms of user interactions. We believe that they are keys to a potential success of the method. In addition to this work, it is also necessary to add the possibility to control the light source by picking on the screen, in particular for the painting tool.

Second, we should reduce the memory footprint. This could be done by using textures with lower resolution but this must be thoroughly studied to ensure that it does not introduce an important bias in the method. Packing values to optimize the bit occupancy could also be a solution to be considered and studied. Concerning the texture duplication due to the Open Inventor

pipeline, it should be removed by fully integrating the method into Open Inventor.

Also, it is important to optimize the computation process to increase interactivity. This implies using a more adapted pattern for the compute groups, which is currently fixed. Indeed, as stated in Brown *et al.* [2012], the efficiency of compute shaders is highly dependent on how the group size is chosen. We did not tackle this problem yet, so this matter should be looked into in the future. Furthermore, some parts of the code, as it is still undergoing development, have not been optimized. Even if these two aspects may not introduce drastic changes in rendering times, they would be interesting.

Finally, we focused on the application for selective diffusion in medical data, thus we have yet to study how this method may be useful for global illumination techniques presented in Section 1.3.3. Due to the various approximations we made, our method can not perform global illumination as well as stochastic techniques. However, it could be used as a tool to design the lighting configuration when rendering volumes. Indeed, due to its capacity to automatically correct itself, the method can withstand many modifications to determine a good configuration.

Conclusion

Summary of the Contributions

In this manuscript, we first focused on light transport and its implication for rendering transparent surfaces and volumes. We also studied how it can be modified to provide additional information using non-photorealistic rendering techniques. In particular, we compared the techniques used in the literature with the features of the Open Inventor SDK.

The conclusions of this study led to the development of a solution to enhance shape depiction for transparent surfaces. Our proposition concerned two aspects of the rendering process. First, to efficiently evaluate derivative information in a 3D space, we design a data structure that allows fast computation of these information. Second, we proposed to modulate the opacity of surfaces using geometric informations like surface curvature. This part of the work was published at the EGSR conference in 2016.

After having addressed the case of transparent surfaces, we tackled the case of volumes. We first studied how our previous solution could extend to volumes, which proved to be adapted to isosurfaces only. By going back to light transport (with the RTE), we tried to modify it to achieve constrained diffusion. We first successfully tested the idea with an extended version of Anisotropic Diffusion. Then we studied different numerical techniques to determine the one best suited to achieve the same kind of results with the RTE. We drew two important conclusions from this study. First, iterative resolution of the steady RTE using a Jacobi method is not stable. Second, solving the unsteady RTE iteratively with a directional basis leads to grid-like artifacts.

Finally, by using the Diffuse Approximation, coupled with the hypothesis of a steady flux, we reduced the problem to a model that has the form of a diffusion equation. This model does not present the same artifacts as the ones we observed during our study, while being stable for a satisfying range of parameters. We then used it to perform selective diffusion by modifying the parameters of the RTE to achieve our goal. Some tools have also been proposed to interact with the diffusion process. However, this work is, at the time of this manuscript, still in development, in particular the visualization and interaction aspects.

Concerning the interaction with Open Inventor, the thesis led to the development of two demonstrators: one for expressive rendering on surfaces (opaque and transparent) and one for computing visibility in volumes (based on Jönsson *et al.* [2012]). A last one is still in development to demonstrate the possibilities of selective diffusion.

Future Work

Surfaces

Toward Global Illumination We have presented a data structure to efficiently access neighbors in a multilayered representation. We think that the interest of this structure is not limited to shape depiction. Indeed, we could extend the amount of optical phenomena that we currently simulate. In particular, it could be used to perform approximated ray intersection and thus, simulate multiple reflections or refractions. Also, efficient access to the real neighborhood, as well as depth information, could be used to approximate sub-surface scattering or translucency with a limited impact on performance.

Third Order Geometric Features In our solution, as we focused on opacity mapping, the only currently supported line rendering features are occluding contours. Thus, we plan to investigate the impact of using line-based rendering, based on inflection points (DeCarlo *et al.* [2003]; Ohtake *et al.* [2004]; Judd *et al.* [2007]; Kolomenkin *et al.* [2008]) to enhance third order features on transparent surfaces.

Note that using third order features also requires to do another derivation pass, which may be too costly to be achieved in real time. Extending to third order features may then require to adapt the Bk-Buffer to store more information and avoid a significant computation overhead.

Support for a User Study The results presented in Chapter 2 are mostly based on internal feedback, thus our conclusion on the impact is probably biased.

However, it can be the support for a user-study to determine the impact on the perception of the shape of the object. In particular, if advanced line rendering is added, it would be interesting to study the impact on legibility.

Volumes

Optimization As stated in Chapter 4, the selective diffusion application needs to be optimized. An aspect that we have yet to consider is to use smaller resolution to compute the diffusion. This would both reduce the memory

impact and the computation cost. However, this implies that the results will probably be less accurate. Also, another solution would be to study the impact of reducing the bit precision of the information, but this presents the same issue as decreasing resolution. Thus, we must conduct a study to determine a good balance between accuracy and efficiency. This step is crucial if we want to provide an efficient solution, and to integrate this application into Open Inventor.

Also, depending on the conclusion of such a study, it could be interesting to use Level of Details (LoD). By using adequately the different levels, we could adapt the computation kernel to provide faster but less accurate results. However, LoD will probably induce a memory overhead compared to using only the top level.

Increasing convergence speed will improve the interactivity of our techniques. This aspect, combined with a reduced memory, should encourage the usage of such an approach.

Visualization As stated in Chapter 4, the algorithm does perform selective diffusion, but the visualization of its results is still limited. As the visualization is an important component of the application, it must be addressed in the future. This will probably require a closer interaction with both our application engineers and specialists in Graphical User Interfaces (GUI).

Also, we should study which quantities (e.g., fluence rate, radiance, gradients, etc) are important in the understanding of the process. Also, we can study how well-defined transfer functions could be useful (Ljung *et al.* [2016]) for this matter, as this aspect was not in the focus of this thesis.

Interaction We introduced many tools to manipulate the algorithm. However, some of them can be enhanced to be more user friendly. In particular, we should add the possibility of interacting with the lighting configuration by picking on the screen (on the slices) directly the positions for the light sources. This would also make the painting tool easier to manipulate and could allow the user to explore more elaborate configurations.

Also, we should propose an intuitive way to choose the different transfer functions involved. For the same purpose, we must ensure affordance in the way the parameters are exposed. This means that we must find a solution so that interacting with the parameters is intuitive, while ensuring that we are within the stability domain of the method.

Toward Global Illumination Due to various approximations we made, our method is limited in the medium and material it can address. Thus, it can not perform global illumination as accurately as stochastic techniques.

However, its main advantage is its capacity to automatically correct itself. Thanks to this, the method can withstand many modifications in terms of pa-

rameters and light configurations (position, intensity...). Thus it could be used for lighting design, to help find a good lighting configuration when rendering volumes.

Also, it could be used for pure surface rendering to perform sub-surface scattering. Indeed, sub-surface scattering is used to approximate light transport in participating media when rendering pure surfaces. By using small patches of voxels on the surfaces, it could be possible to use our algorithm to evaluate this scattering.

Software Contributions

In the additional chapter, we give an overview of the different contributions that have been proposed for Open Inventor. First we present the applications that focus on surfaces, then a demonstration for visibility in volumes that originated from state-of-the-art techniques, and finally, the current state of the prototype for selective diffusion.

Demonstrator for Surfaces

The demonstration for surfaces is composed of two modules: one for opaque surfaces with state-of-the-art physically plausible as well as expressive rendering technique, and one that extends to transparent surfaces.

Opaque surfaces and lighting

A first contribution for Open Inventor was developed in the early work of this thesis. It is mainly existing techniques, adapted to the Open Inventor pipeline. Here is a list of the features that were proposed:

- Deferred renderer with two passes of derivation. This feature, based on an existing Open Inventor node, operates two deferred rendering passes to compute an order of derivative per pass.
- Image-based Lighting (IBL) using Ramamoorthi and Hanrahan [2001] for Lambertian BRDF and averaged mipmaps to approximate Prefiltered Environment Maps.
- HDR tone mapping (Hable [2010]) to account for HDR environment maps.
- Physically-plausible BRDF model (using Burley and Walt Disney Animation [2012]).
- Screen-space STAR expressive rendering technique for opaque surfaces (Vergne *et al.* [2010], Kolomenkin *et al.* [2008]).
- Screen-Space Ambient Occlusion (Bavoil *et al.* [2008]).

The light model and IBL features have been integrated in the VolumeViz extension (release version 9.8 of Open Inventor) and presented at the Radiological Society of North America (RSNA) during its 2016 exposition. It should

be integrated for surfaces as well in a future version (10.x) of Open Inventor.

Transparent surfaces visualization

A second contribution was centered around transparent surfaces to propose an opacity modulation based on geometric features. Here is a list of the features that were proposed:

- Transparency group to handle the Bk-Buffer. This group is responsible of managing the shaders and buffers to operate the Bk-Buffer
- Adaptation of existing techniques designed for opaque surface to transparent one (Kindlmann *et al.* [2003], Vergne *et al.* [2010]).
- Opacity modulation using geometric information (Murray *et al.* [2016]).

This work was presented at the EuroGraphics Symposium on Rendering (EGSR) in June 2016 in Dublin, in the Experimental Ideas & Implementations track (Murray *et al.* [2016]). It has yet to be integrated into Open Inventor.

Visibility for Volumes

This third contribution is extracted from our research while studying the literature. The goal is to propose an enhanced shadowing option compared to the ones currently available in Open Inventor. Here is a list of the features that were proposed:

- Compute Shader pipeline in addition to the graphics pipeline in Open Inventor. This feature is necessary to compute the visibility factor and can obviously be used for any other computing purpose.
- Implementation of a method derived from the visibility techniques from Ritschel [2007] and Jönsson *et al.* [2013].

The demonstration was presented at the RSNA exposition in 2017. It has yet to be integrated into Open Inventor.

Selective Diffusion

This last contribution is still in development. Compared to other contributions, this one relies mostly on Open Inventor features. Indeed, except for the compute shader pipeline presented above which we use to compute diffusion, everything else is implemented using Open Inventor (and VolumeViz) existing nodes.

However, due to its development state, it has not been presented yet.

Appendix A

Steady RTE Expressed with Finite Elements

This chapter presents the mathematical details for the expression of the steady RTE using the Finite Element Method.

The steady RTE with an Emission-Absorption model can be expressed as the linear system:

$$ML = Q_e - RL$$

The weak form of the steady RTE, Equation 3.15, is recalled below:

$$\begin{aligned}\sum_j \sum_l L_{j,l} \int_{4\pi,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\vec{\omega}^T \cdot \vec{\nabla}_p(\psi_j(\vec{\omega})\varphi_l(p)) = \\ \int_{4\pi,[-1,1]} \psi_j(\vec{\omega})\varphi_l(p)Q_e(p) \\ -\sum_j \sum_l L_{j,l} \int_{4\pi,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\psi_j(\vec{\omega})\varphi_l(p)K_t(p)\end{aligned} \tag{A.1}$$

A.1 Mathematical Details for M

The left side of Equation A.1 represent the terms of the matrix M:

$$M_{i,j,k,l} = \int_{4\pi,[-1,1]} \psi_i(\vec{\omega})\varphi_k(p)\vec{\omega}^T \cdot \vec{\nabla}_p(\psi_j(\vec{\omega})\varphi_l(p)) \tag{A.2}$$

In Equation A.2, we can separate the position-dependent integral from the direction-dependent one, as the gradient is only a spatial one. Then, we obtain Equation A.3.

$$M_{i,j,k,l} = \int_{4\pi} \psi_i(\vec{\omega})\psi_j(\vec{\omega})\vec{\omega}^T \cdot \int_{[-1,1]} \varphi_k(p)\vec{\nabla}_p\varphi_l(p) \tag{A.3}$$

We will use the notations $M_{i,j,k,l} = \vec{\alpha}_{i,j}^T \cdot \vec{\beta}_{k,l}$ for clarity.

A.1.1 Computing $\vec{\alpha}_{i,j}$

Computing $\vec{\alpha}_{i,j} = \int_{4\pi} \psi_i(\vec{\omega})\psi_j(\vec{\omega})\vec{\omega}\partial\vec{\omega}$ consists in computing 3 4×4 matrices (one for each component of $\vec{\omega}$):

$$\begin{bmatrix} x & x^2 & xy & xz \\ x^2 & x^3 & x^2y & x^2z \\ xy & x^2y & xy^2 & xyz \\ xz & x^2z & xyz & xz^2 \end{bmatrix}; \begin{bmatrix} y & xy & y^2 & yz \\ xy & x^2y & xy^2 & xyz \\ y^2 & xy^2 & y^3 & y^2z \\ yz & xyz & y^2z & yz^2 \end{bmatrix}; \begin{bmatrix} z & xz & yz & z^2 \\ xz & x^2z & xyz & xz^2 \\ yz & xyz & y^2z & yz^2 \\ z^2 & xz^2 & yz^2 & z^3 \end{bmatrix}$$

For each term of these matrices, we must evaluate the integral for all directions. We will use the spherical notation:

$$x = \sin(\theta)\sin(\phi)$$
$$y = \sin(\theta)\cos(\phi)$$
$$z = \cos(\theta)$$
$$\int_{4\pi} \partial\vec{\omega} = \int_{\pi}\int_{2\pi} \sin(\theta)\partial\theta\partial\phi$$

The result is the following:

$$\vec{\alpha}_{(i,j)\in[0,3]^2} = \begin{bmatrix} (0,0,0) & (\frac{4\pi}{3},0,0) & (0,\frac{4\pi}{3},0) & (0,0,\frac{4\pi}{3}) \\ (\frac{4\pi}{3},0,0) & (0,0,0) & (0,0,0) & (0,0,0) \\ (0,\frac{4\pi}{3},0) & (0,0,0) & (0,0,0) & (0,0,0) \\ (0,0,\frac{4\pi}{3}) & (0,0,0) & (0,0,0) & (0,0,0) \end{bmatrix} \tag{A.4}$$

The important number of zeros is interesting as it implies that computing M will not require too many operations.

A.1.2 Computing $\vec{\beta}_{k,l}$

For this part, we will compute separately the components of $\vec{\beta}_{k,l}$. Furthermore, the position p can be decomposed, as x, y and z are independent, and we can write:

$$\varphi_k(p) = \varphi_{k_x}(x)\varphi_{k_y}(y)\varphi_{k_z}(z)$$

Also, the gradient ∇_p is decomposed into (d_x,d_y,d_z) and thus:

$$\vec{\beta}_{k,l} = \begin{pmatrix} \int_{[-1,1]} \varphi_{k_x}(x)\mathrm{d}_x\varphi_{l_x}(x) \int_{[-1,1]} \varphi_{k_y}(y)\varphi_{l_y}(y) \int_{[-1,1]} \varphi_{k_z}(z)\varphi_{l_z}(z) \\ \int_{[-1,1]} \varphi_{k_x}(x)\varphi_{l_x}(x) \int_{[-1,1]} \varphi_{k_y}(y)\mathrm{d}_y\varphi_{l_y}(y) \int_{[-1,1]} \varphi_{k_z}(z)\varphi_{l_z}(z) \\ \int_{[-1,1]} \varphi_{k_x}(x)\varphi_{l_x}(x) \int_{[-1,1]} \varphi_{k_y}(y)\varphi_{l_y}(y) \int_{[-1,1]} \varphi_{k_z}(z)\mathrm{d}_z\varphi_{l_z}(z) \end{pmatrix}$$

To compute $\vec{\beta}_{k,l}$, we recall the formulation of the basis φ.

$$\varphi_k(p) = \begin{cases} 1+p & \text{if } p_{k-1} \le p \le p_k \\ 1-p & \text{if } p_k \le p \le p_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi'_k(p) = \begin{cases} 1 & \text{if } p_{k-1} \leq p \leq p_k \\ -1 & \text{if } p_k \leq p \leq p_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

Thus, the products of φ give:

$$\int_{[-1,1]} \varphi_k(p)\varphi'_l(p) = \begin{cases} \int_{[-1,0]}(1+p)\cdot(-1) + \int_{[0,1]}(1-p)\cdot 0 & \text{if } l = k-1 \\ \int_{[-1,0]}(1+p)\cdot 0 + \int_{[0,1]}(1-p)\cdot 1 & \text{if } l = k+1 \\ \int_{[-1,0]}(1+p)\cdot 1 + \int_{[0,1]}(1-p)\cdot(-1) & \text{if } l = k \\ 0 & \text{if } |l-k| > 1 \end{cases}$$

$$\int_{[-1,1]} \varphi_k(p)\varphi'_l(p) = \begin{cases} \frac{1}{2} + 0 & \text{if } l = k-1 \\ 0 + \frac{-1}{2} & \text{if } l = k+1 \\ \frac{1}{2} + \frac{-1}{2} & \text{if } l = k \\ 0 & \text{if } |l-k| > 1 \end{cases}$$

$$\int_{[-1,1]} \varphi_k(p)\varphi_l(p) = \begin{cases} \int_{[-1,0]}(1+p)\cdot(-p) + \int_{[0,1]}(1-p)\cdot 0 & \text{if } l = k-1 \\ \int_{[-1,0]}(1+p)\cdot 0 + \int_{[0,1]}(1-p)\cdot p & \text{if } l = k+1 \\ \int_{[-1,0]}(1+p)^2 + \int_{[0,1]}(1-p)^2 & \text{if } l = k \\ 0 & \text{if } |l-k| > 1 \end{cases}$$

$$\int_{[-1,1]} \varphi_k(p)\varphi_l(p) = \begin{cases} \frac{1}{6} + 0 & \text{if } l = k-1 \\ 0 + \frac{1}{6} & \text{if } l = k+1 \\ \frac{1}{3} + \frac{1}{3} & \text{if } l = k \\ 0 & \text{if } |l-k| > 1 \end{cases}$$

Note that $\varphi_k(p) = 1 + p$ becomes $\varphi_k(p) = p$ when shifted from p to $p + 1$ and $\varphi_k(p) = 1 - p$ becomes $\varphi_k(p) = -p$ when shifted from p to $p - 1$. This corresponds to the cases $|l - k| = 1$.

We can also note that all three component of $\vec{\beta}_{k,l}$ are equal:

$$\vec{\beta}_{k,l} = (\beta_{k,l}, \beta_{k,l}, \beta_{k,l})$$

For a practical reason, the formulation we use must be adapted to a 3D volume(of size $W \times H \times D$): $(k,l) \in [0,N]^2 \iff ((k_x,k_y,k_z),(l_x,l_y,l_z)) \in ([0,W],[0,H],[0,D])^2$. That way, we can express the system as a reduced kernel that is computed for a voxel at position k. As for all cases were $|l - k| > 1$, the matrix will contains zeros, we can also reduce (k,l) to $k + d, d \in \{-1,0,1\}^3$. Thus, the result for $\vec{\beta}_{k,l}$ are the following coefficients, that must be applied to the neighborhood, for each axis (x, y and z):

$$\beta_{d\in\{-1,0,1\}^3} = \frac{1}{9}\left[\begin{bmatrix} \frac{-1}{8} & 0 & \frac{1}{8} \\ \frac{-1}{2} & 0 & \frac{1}{2} \\ \frac{-1}{8} & 0 & \frac{1}{8} \end{bmatrix}\begin{bmatrix} \frac{-1}{2} & 0 & \frac{1}{2} \\ -2 & 0 & 2 \\ \frac{-1}{2} & 0 & \frac{1}{2} \end{bmatrix}\begin{bmatrix} \frac{-1}{8} & 0 & \frac{1}{8} \\ \frac{-1}{2} & 0 & \frac{1}{2} \\ \frac{-1}{8} & 0 & \frac{1}{8} \end{bmatrix}\right] \tag{A.5}$$

A.2 Mathematical details for R

By using the same reasoning, the coefficients of the matrix R will be the one in Equation A.6.

$$R_{i,j,k,l} = \rho \int_{4\pi} \psi_i(\vec{\omega})\psi_j(\vec{\omega}) \int_{[-1,1]} \varphi_k(p)\varphi_l(p) \tag{A.6}$$

We will use the notations $K_{i,j,k,l} = K_t\gamma_{i,j}\delta_{k,l}$ for clarity.

A.2.1 Computing $\gamma_{i,j}$

Computing $\gamma_{i,j}$ consists in computing the coefficients of the following matrix:

$$\begin{bmatrix} 1 & x & y & z \\ x & x^2 & xy & xz \\ y & xy & y^2 & yz \\ z & xz & yz & z^2 \end{bmatrix}$$

As we have already calculated all the above terms when we evaluated $\vec{\alpha}_{i,j}$, the result here can immediately be assumed:

$$\gamma_{i,j} = \begin{bmatrix} 4\pi & 0 & 0 & 0 \\ 0 & \frac{4\pi}{3} & 0 & 0 \\ 0 & 0 & \frac{4\pi}{3} & 0 \\ 0 & 0 & 0 & \frac{4\pi}{3} \end{bmatrix} \tag{A.7}$$

Once again, there are many zeros in this matrix.

A.2.2 Computing $\delta_{k,l}$

Computing $\delta_{k,l}$ is also straightforward. Indeed, as x, y and z are independent, we still have:

$$\varphi_k(p) = \varphi_{k_x}(x)\varphi_{k_y}(y)\varphi_{k_z}(z)$$

Thus:

$$\delta_{k,l} = \int_{[-1,1]} \varphi_{k_x}(x)\varphi_{l_x}(x) \int_{[-1,1]} \varphi_{k_y}(y)\varphi_{l_y}(y) \int_{[-1,1]} \varphi_{k_z}(z)\varphi_{l_z}(z)$$

The different possible values have already been calculated for $\vec{\beta}_{k,l}$. So finally, to compute $\delta_{k,l}$, and using the same adaptation to 3D volumes as for $\beta_{k,l}$, the following coefficients must be applied to the neighborhood:

$$\delta_{d\in\{-1,0,1\}^3} = \left[\begin{bmatrix} \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \end{bmatrix} \begin{bmatrix} \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{2}{27} & \frac{8}{27} & \frac{2}{27} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \end{bmatrix} \begin{bmatrix} \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \\ \frac{1}{54} & \frac{2}{27} & \frac{1}{54} \\ \frac{1}{216} & \frac{1}{54} & \frac{1}{216} \end{bmatrix} \right] \tag{A.8}$$

Appendix B

The Order Moment Method in Details

This chapter presents the mathematical details to obtain our different moment-based models. We present the system obtained by using up to the first order, then we present the system when considering up to the second order.

B.1 System with First Order Moment

B.1.1 Order 0 (μ_0)

$L(p, \vec{\omega}, t)$**:**

For order 0, two terms are straightforward: $L(p, \vec{\omega}, t)$ and $\frac{\partial L(p,\vec{\omega},t)}{\partial t}$. By definition, order 0 of these two terms is the fluence rate:

$$\boxed{\begin{array}{c} \mu_0(L(p, \vec{\omega}, t)) = \phi(p, t) \\ \mu_0(\dfrac{\partial L(p, \vec{\omega}, t)}{\partial t}) = \dfrac{\partial \phi(p, t)}{\partial t} \end{array}} \tag{B.1}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p, \vec{\omega}, t)$**:**

For this term, we have to proceed to a little reorganization:

$$\int_{4\pi} \vec{\omega}^T \cdot \vec{\nabla}_p L(p, \vec{\omega}, t) d\omega = \vec{\nabla}_p{}^T \int_{4\pi} \vec{\omega} L(p, \vec{\omega}, t) d\omega$$

Thus, by definition, we have:

$$\mu_0(\vec{\omega} \cdot \vec{\nabla}_p L(p, \vec{\omega}, t)) = \vec{\nabla}_p{}^T \mu_1(L(p, \vec{\omega}, t))$$

So finally:

$$\boxed{\mu_0(\vec{\omega} \cdot \vec{\nabla}_p L(p, \vec{\omega}, t)) = \vec{\nabla}_p{}^T \vec{E}(p, t)} \tag{B.2}$$

$\mathcal{P}(\vec{\omega}, \vec{\omega_i}) L_i(p, \vec{\omega_i}, t)$:

Here we have to evaluate:

$$\int_{4\pi} \left(\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) L_i(p, \vec{\omega_i}, t) d\omega_i \right) d\omega$$

As only $\mathcal{P}$ depends on $\vec{\omega}$, we can write:

$$\int_{4\pi} \left(\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) d\omega \right) L_i(p, \vec{\omega_i}, t) d\omega_i$$

By definition, we have $\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) d\omega = 1$, thus:

$$\int_{4\pi} \left(\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) L_i(p, \vec{\omega_i}, t) d\omega_i \right) d\omega = \int_{4\pi} L_i(p, \vec{\omega_i}, t) d\omega_i$$

We now have two possibilities in the interpretation of L_i:

- L_i is the radiance emitted by a neighboring element at $p + \epsilon\vec{\omega_i}$ toward the current one at p. In this case: $L_i(p, \vec{\omega_i}, t) \equiv L(p + \epsilon\vec{\omega_i}, -\vec{\omega_i}, t)$.
- L_i is the radiance received at p from the direction $\vec{\omega_i}$. In this case: $L_i(p, \vec{\omega_i}, t) \equiv L(p, \vec{\omega_i}, t)$.

In the first case, we cannot simplify any further $\int_{4\pi} L_i(p, \vec{\omega_i}, t) d\omega_i$. However, in the second case, the integral corresponds to the definition of the fluence rate and thus:

$$\boxed{\mu_0 \left(\int_{4\pi} \mathcal{P}(\vec{\omega}, \vec{\omega_i}) L_i(p, \vec{\omega_i}, t) d\omega_i \right) = \phi(p, t)} \tag{B.3}$$

B.1.2 Order 1 (μ_1)

$L(p, \vec{\omega}, t)$:

For order 1, $L(p, \vec{\omega}, t)$ and $\frac{\partial L(p, \vec{\omega}, t)}{\partial t}$ are also straightforward. By definition, order 1 of these two terms is the irradiance vector:

$$\boxed{\begin{aligned} \mu_1(L(p, \vec{\omega}, t)) &= \vec{E}(p, t) \\ \mu_1(\frac{\partial L(p, \vec{\omega}, t)}{\partial t}) &= \frac{\partial \vec{E}(p, t)}{\partial t} \end{aligned}} \tag{B.4}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p, \vec{\omega}, t)$:

For this term, we have:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p L(p, \vec{\omega}, t) d\omega = \vec{\nabla}_p \mu_2(L(p, \vec{\omega}, t))$$

Note that $\mu_2(L(p,\vec{\omega},t))$ is a symmetric tensor that corresponds to the radiative pressure. In the diffuse approximation, this tensor is isotropic and simply becomes $\frac{1}{3}\phi(p,t)I$.

This result can also be obtained by using directly the diffuse approximation:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p L(p,\vec{\omega},t)d\omega = \int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p (\frac{1}{4\pi}\phi(p,t) + \frac{3}{4\pi}\vec{\omega}^T \vec{E}(p,t))d\omega$$

For the part on ϕ, as $\vec{\omega}\vec{\omega}^T$ is a symmetric matrix, we can use the equivalence:

$$\vec{\omega}\vec{\omega}^T \cdot \vec{\nabla}_p \phi(p,t) = [(\vec{\nabla}_p \phi(p,t))^T \cdot \vec{\omega}\vec{\omega}^T]^T$$

As ϕ does not depend on $\vec{\omega}$, we can extract it from the integral:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p \phi(p,t)d\omega = [\vec{\nabla}_p{}^T \phi(p,t) \int_{4\pi} \vec{\omega}\vec{\omega}^T d\omega]^T$$

with (see Appendix A.1 for the simplifications):

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T d\omega = \frac{4\pi}{3} I$$

which gives us:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p \phi(p,t)d\omega = \frac{4\pi}{3}\vec{\nabla}_p \phi(p,t)$$

For $\vec{E}(p,t)$, the formulation is a vector containing terms depending on : $\omega.x^i * \omega.y^j \omega.z^k$ with $i+j+k=3$, which integrates to 0. Thus we have:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T \vec{E}(p,t))d\omega = (0)$$

Either way, we finally obtain:

$$\boxed{\mu_1(\vec{\omega} \cdot \vec{\nabla}_p L(p,\vec{\omega},t)) = \frac{1}{3}\vec{\nabla}_p \phi(p,t)} \tag{B.5}$$

$\mathcal{P}(\vec{\omega},\vec{\omega_i})L_i(p,\vec{\omega_i},t)$**:**

Here we have to evaluate:

$$\int_{4\pi} \vec{\omega} \left(\int_{4\pi} \mathcal{P}(\vec{\omega},\vec{\omega_i}) L_i(p,\vec{\omega_i},t)d\omega_i \right) d\omega$$

To simplify this term, we need to use the same consideration on L_i as the one used previously: $L_i(p,\vec{\omega_i},t) \equiv L(p,\vec{\omega_i},t)$. This way, we can write:

$$\int_{4\pi} \vec{\omega} \left(\int_{4\pi} \mathcal{P}(\vec{\omega},\vec{\omega_i}) L_i(p,\vec{\omega_i},t)d\omega_i \right) d\omega =$$
$$\int_{4\pi} \vec{\omega} \left(\int_{4\pi} \mathcal{P}(\vec{\omega},\vec{\omega_i}) (\frac{1}{4\pi}\phi(p,t) + \frac{3}{4\pi}\vec{\omega_i} \cdot \vec{E}(p,t))d\omega_i \right) d\omega$$

We can now separate the two terms (ϕ and $\vec{E}$). For ϕ, we have:

$$\frac{1}{4\pi}\phi(p,t)\int_{4\pi}\vec{\omega}\left(\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})d\omega_i\right)d\omega$$

By definition, $\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})d\omega_i = 1$ and $\int_{4\pi}\vec{\omega}d\omega = \vec{0}$:

$$\frac{1}{4\pi}\phi(p,t)\int_{4\pi}\vec{\omega}\left(\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})d\omega_i\right)d\omega = \vec{0}$$

For $\vec{E}$, we have:

$$\frac{3}{4\pi}\int_{4\pi}\vec{\omega}\left(\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})(\vec{\omega_i}\cdot\vec{E}(p,t))d\omega_i\right)d\omega$$

which is similar to:

$$\frac{3}{4\pi}\int_{4\pi}\vec{\omega}\left(\left(\int_{4\pi}\vec{\omega_i}\mathcal{P}(\vec{\omega},\vec{\omega_i})d\omega_i\right)\cdot\vec{E}(p,t)\right)d\omega$$

Pierrat [2007] and Carminati [2016] proved that:

$$\int_{4\pi}\vec{\omega_i}\mathcal{P}(\vec{\omega},\vec{\omega_i})d\omega_i = g\vec{\omega}$$

As $\vec{\omega}(\vec{\omega}^T\vec{E}(p,t)) = \left(\vec{E}^T(p,t)(\vec{\omega}\vec{\omega}^T)\right)^T$, which integrates to $\frac{4\pi}{3}\vec{E}(p,t)$, we finally have:

$$\frac{3}{4\pi}\int_{4\pi}\vec{\omega}\left(\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})(\vec{\omega_i}\cdot\vec{E}(p,t))d\omega_i\right)d\omega = g\vec{E}(p,t)$$

Finally, we obtain:

$$\boxed{\mu_1\left(\int_{4\pi}\mathcal{P}(\vec{\omega},\vec{\omega_i})L_i(p,\vec{\omega_i},t)d\omega_i\right) = g\vec{E}(p,t)} \tag{B.6}$$

B.1.3 Final System

By combining all the results presented above, we can finally obtain our linear system, described by Equations B.7 and B.8, with $K_{t'} = K_a + (1-g)K_s$.

$$\frac{1}{c}\frac{\partial\phi(p,t)}{\partial t} = -div(\vec{E}(p,t)) - K_a(p)\phi(p,t) + \phi_e(p,t) \tag{B.7}$$

$$\frac{1}{c}\frac{\partial\vec{E}(p,t)}{\partial t} = -\frac{1}{3}\vec{\nabla}_p\phi(p,t) - K_{t'}(p)\vec{E}(p,t) + \vec{E}_e(p,t) \tag{B.8}$$

B.2 System with Second Order Moment

We will detail here the computation of the first and second order moments with the following radiance decomposition:

$$L(p,\vec{\omega},t) = \vec{\omega}^T \cdot \vec{E}(p,t) + \vec{\omega}^T S(p,t)\vec{\omega}$$

where $\vec{E}$ is the first moment of the radiance, and S its second order moment. S is a symmetric tensor linked to its second moment, but is not exactly the second moment as the moment of order 0 is embedded in its trace.

As a reminder, the RTE is:

$$\begin{aligned}\frac{1}{c}\frac{\partial L(p,t,\vec{\omega})}{\partial t} + \vec{\omega}^T \cdot \vec{\nabla}_p L(p,t,\vec{\omega}) = -K_t(p)\cdot L(p,t,\vec{\omega}) + Q_e(p,t,\vec{\omega}) \\ + \int_{4\pi} K_s(p)\cdot \mathcal{P}(p,\vec{\omega},\vec{\omega}')\cdot L_i(p,t,\vec{\omega}')d\vec{\omega}'\end{aligned} \tag{B.9}$$

B.2.1 First Order Moment μ_1

The first order moment is obtained by projecting the RTE on $\vec{\omega}$ and then integrating over the sphere ($\int_{4\pi}(RTE)\vec{\omega}d\omega$).

In the RTE, we have two kinds of terms, those based on $L(p,t,\vec{\omega})$ (including the time derivative) and the directional derivative $\vec{\omega}\cdot\vec{\nabla}_p L(p,t,\vec{\omega})$.

For the first ones, the first moment is straightforward, it is by definition $\vec{E}$. The gradient, however, needs to be developed. To do so, we will decompose it according to our radiance decomposition and define $\vec{\omega} = (x,y,z)$.

$L(p,t,\vec{\omega})$:

By definition, we directly have:

$$\boxed{\begin{aligned}\mu_1(L(p,t,\vec{\omega})\vec{\omega}) &= \frac{4\pi}{3}\vec{E}(p,t) \\ \mu_1\left(\frac{\partial L(p,t,\vec{\omega})}{\partial t}\vec{\omega}\right) &= \frac{4\pi}{3c}\frac{\partial \vec{E}(p,t)}{\partial t}\end{aligned}} \tag{B.10}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p,\vec{\omega},t)$:

For this term, we have:

$$\int_{4\pi} \vec{\omega}\vec{\omega}^T \vec{\nabla}_p L(p,\vec{\omega},t)d\omega = \vec{\nabla}_p \mu_2(L(p,\vec{\omega},t))$$

As we have embedded the moment of order 0 into the term of order 2, we can not use directly the radiative pressure here. Thus, we use the radiance decomposition to solve this term.

First we focus on the term depending on $\vec{E}$. Thus we need to evaluate:

$$\int_{4\pi} \vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T \vec{E}(p,t)) \vec{\omega} d\omega = (\vec{\nabla}_p^T \int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T \vec{E}(p,t)) d\omega)^T$$

where:

$$(\vec{\omega}\vec{\omega}^T) = \begin{pmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{pmatrix}$$

Thus $(\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T \vec{E}(p,t))$ is a matrix containing terms depending on $x^i y^j z^k$ with $i+j+k=3$. As stated before in Appendix A, all the combinations of $x^i y^j z^k$ with $i+j+k=3$ integrate to 0 on the sphere. As $\vec{E}(p,t)$ does not depend on $\vec{\omega}$, the result is:

$$\boxed{\int_{4\pi} \vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T \vec{E}(p,t)) \vec{\omega} d\omega = \vec{0}} \tag{B.11}$$

Now we focus on the term depending on S. Thus we need to evaluate:

$$\int_{4\pi} \vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T S(p,t) \vec{\omega}) \vec{\omega} d\omega = (\vec{\nabla}_p^T \int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T S(p,t)\vec{\omega}) d\omega)^T$$

Once again, we have:

$$(\vec{\omega}\vec{\omega}^T) = \begin{pmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{pmatrix}$$

and as S is symmetric, $S_{0,1} = S_{1,0}$, $S_{0,2} = S_{2,0}$, $S_{1,2} = S_{2,1}$, we have:

$$\vec{\omega}^T S(p,t) \vec{\omega} = x^2 S_{0,0} + y^2 S_{1,1} + z^2 S_{2,2} + 2xy S_{0,1} + 2xz S_{0,2} + 2yz S_{1,2}$$

Thus, the result will be a vector with a combination of terms depending on $x^i y^j z^k$ with $i+j+k=4$. Using the same notation as for the $\vec{E}$ part, we obtain that the terms where $i|j|k = 3$ integrate to 0 over the sphere. The terms where $i|j|k = 4$ integrate to $\frac{4\pi}{5}$ and the rest integrate to $\frac{4\pi}{15}$. The details for this integration is to long to appear here. The evaluation of the integral term can be computed with an analytic solver such as Maxima.

Thus, we have:

$$\int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T S(p,t)\vec{\omega}) d\omega$$
$$= \frac{4\pi}{5} \begin{bmatrix} S_{0,0} + \frac{1}{3}(S_{1,1} + S_{2,2}) & \frac{2}{3} S_{0,1} & \frac{2}{3} S_{0,2} \\ \frac{2}{3} S_{0,1} & S_{1,1} + \frac{1}{3}(S_{0,0} + S_{2,2}) & \frac{2}{3} S_{1,2} \\ \frac{2}{3} S_{0,2} & \frac{2}{3} S_{1,2} & S_{2,2} + \frac{1}{3}(S_{0,0} + S_{1,1}) \end{bmatrix}$$

The expression can be slightly simplified as $S_{0,0} + S_{1,1} + S_{2,2}$ is the trace of the matrix S ($tr(S)$), allowing us to regroup some terms:

$$\int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T S(p,t)\vec{\omega}) d\omega = \frac{4\pi}{5} \frac{1}{3} (2S + tr(S) I)$$

$$\boxed{\int_{4\pi} \vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T S(p,t)\vec{\omega})\vec{\omega} d\omega = \frac{4\pi}{5}\frac{1}{3}\begin{pmatrix} 2\delta_x S_{0,0} + \delta_x tr(S) + 2(\delta_y S_{0,1} + \delta_z S_{0,2}) \\ 2\delta_y S_{1,1} + \delta_y tr(S) + 2(\delta_x S_{0,1} + \delta_z S_{1,2}) \\ 2\delta_z S_{2,2} + \delta_z tr(S) + 2(\delta_x S_{0,2} + \delta_y S_{1,2}) \end{pmatrix}} \tag{B.12}$$

B.2.2 Second-Order Moment μ_2

The second-order moment is obtained by multiplying the RTE by the matrix $\vec{\omega}\vec{\omega}^T$ and then integrating over the sphere ($\int_{4\pi}(RTE)(\vec{\omega}\vec{\omega}^T)d\omega$).

As S is not exactly the second moment of the radiance, we do not have an immediate simplification of the 0 order space derivative terms. However, the terms coming from the first order moment ($\vec{E}$) disappear when computing the second order moment ($i+j+k=3$). Thus, all that is left is:

$$\int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T S(p,t)\vec{\omega}) d\omega$$

which by chance, has already been calculated in the previous section:

$$\int_{4\pi} (\vec{\omega}\vec{\omega}^T)(\vec{\omega}^T S(p,t)\vec{\omega}) d\omega = \frac{4\pi}{5}\frac{1}{3}(2S + tr(S)I)$$

Thus, these terms will be the following.

$L(p,t,\vec{\omega})$:

$$\boxed{\begin{aligned} \mu_2(L(p,t,\vec{\omega})) &= \frac{4\pi}{15}(2S(p,t) + tr(S(p,t))I) \\ \mu_2\left(\frac{\partial L(p,t,\vec{\omega})}{\partial t}\right) &= \frac{4\pi}{15}(2\frac{\partial S(p,t)}{\partial t} + \frac{\partial tr(S(p,t))}{\partial t}I) \end{aligned}} \tag{B.13}$$

$\vec{\omega}^T \cdot \vec{\nabla}_p L(p,\vec{\omega},t)$:

As for the first moment, we will evaluate this term by using the radiance decomposition.

First we focus on the term depending on $\vec{E}$. Thus we need to evaluate:

$$\int_{4\pi} (\vec{\omega}\vec{\omega}^T)\vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T \vec{E}(p,t)) d\omega$$

For this term, we need a little development:

$$\vec{\omega}^T \vec{\nabla}_p (\vec{\omega}^T \vec{E}(p,t)) = \begin{pmatrix} x^2\delta_x E_0 + xy\delta_x E_1 + xz\delta_x E_2 \\ xy\delta_y E_0 + y^2\delta_y E_1 + yz\delta_y E_2 \\ xz\delta_z E_0 + yz\delta_z E_1 + z^2\delta_z E_2 \end{pmatrix}$$

Thus the term $(\vec{\omega}\vec{\omega}^T)\vec{\omega}^T\vec{\nabla}_p(\vec{\omega}^T\vec{E}(p,t))$ leads to a combination of terms depending on $x^iy^jz^k$ with $i+j+k=4$. Same as before, we obtain that the terms where $i|j|k=3$ integrate to 0 over the sphere. The terms where $i|j|k=4$ integrate to $\frac{4\pi}{5}$ and the rest integrate to $\frac{4\pi}{15}$. Thus, what we have left after integration is:

$$\frac{4\pi}{5}\begin{bmatrix} \delta_xE_0+\frac{1}{3}(\delta_yE_1+\delta_zE_2) & \frac{1}{3}(\delta_xE_1+\delta_yE_0) & \frac{1}{3}(\delta_xE_2+\delta_zE_0) \\ \frac{1}{3}(\delta_xE_1+\delta_yE_0) & \delta_yE_1+\frac{1}{3}(\delta_xE_0+\delta_zE_2) & \frac{1}{3}(\delta_yE_2+\delta_zE_1) \\ \frac{1}{3}(\delta_xE_2+\delta_zE_0) & \frac{1}{3}(\delta_yE_2+\delta_zE_1) & \delta_zE_2+\frac{1}{3}(\delta_xE_0+\delta_yE_1) \end{bmatrix}$$

As $\delta_xE_0+\delta_yE_1+\delta_zE_2 = div(\vec{E})$ we can reformulate into a more compact term:

$$\begin{aligned}\mu_2(\vec{\omega}^T\cdot\vec{\nabla}_p(\vec{\omega}^T\vec{E}(p,t)) &= \frac{4\pi}{15}(div(\vec{E})I \\ &+ \begin{bmatrix} 2\delta_xE_0 & (\delta_xE_1+\delta_yE_0) & (\delta_xE_2+\delta_zE_0) \\ (\delta_xE_1+\delta_yE_0) & 2\delta_yE_1 & (\delta_yE_2+\delta_zE_1) \\ (\delta_xE_2+\delta_zE_0) & (\delta_yE_2+\delta_zE_1) & 2\delta_zE_2 \end{bmatrix})\end{aligned} \tag{B.14}$$

Now we focus on the term depending on S. We want to evaluate:

$$\int_{4\pi}(\vec{\omega}\vec{\omega}^T)\vec{\omega}^T\vec{\nabla}_p(\vec{\omega}^TS(p,t)\vec{\omega})d\omega$$

Using the same ideas as the one used to establish Equation B.12, we can obtain a combination of terms depending on $x^iy^jz^k$ with $i+j+k=5$. Using spherical notations, these terms integrate to 0 for any combination.

Thus, this term reduces to a null matrix.

$$\int_{4\pi}(\vec{\omega}\vec{\omega}^T)\vec{\omega}^T\vec{\nabla}_p(\vec{\omega}^TS(p,t)\vec{\omega})d\omega = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{B.15}$$

Finally, this gives us:

$$\boxed{\begin{aligned}\mu_2(\vec{\omega}^T\vec{\nabla}_pL(p,t,\vec{\omega})) &= \frac{4\pi}{15}(div(\vec{E})I \\ &+ \begin{bmatrix} 2\delta_xE_0 & (\delta_xE_1+\delta_yE_0) & (\delta_xE_2+\delta_zE_0) \\ (\delta_xE_1+\delta_yE_0) & 2\delta_yE_1 & (\delta_yE_2+\delta_zE_1) \\ (\delta_xE_2+\delta_zE_0) & (\delta_yE_2+\delta_zE_1) & 2\delta_zE_2 \end{bmatrix})\end{aligned}} \tag{B.16}$$

B.2.3 Final System

Once assembled, the different terms lead us to the following system:

$$\frac{1}{c}\frac{\partial \vec{E}(p,t)}{\partial t} + \frac{1}{5}\begin{pmatrix} \delta_x tr(S) + 2\delta_x S_{xx} + 2(\delta_y S_{xy} + \delta_z S_{xz}) \\ \delta_y tr(S) + 2\delta_y S_{yy} + 2(\delta_x S_{xy} + \delta_z S_{yz}) \\ \delta_z tr(S) + 2\delta_z S_{zz} + 2(\delta_x S_{xz} + \delta_y S_{yz}) \end{pmatrix} = \vec{E}_e(p,t) - K_t(p)\vec{E}(p,t)$$

$$\frac{1}{c}\frac{\partial}{\partial t}(2S(p) + tr(S(p))I) + \begin{bmatrix} 2\delta_x E_x + div(\vec{E}) & (\delta_x E_y + \delta_y E_x) & (\delta_x E_z + \delta_z E_x) \\ (\delta_x E_y + \delta_y E_x) & 2\delta_y E_y + div(\vec{E}) & (\delta_y E_z + \delta_z E_y) \\ (\delta_x E_z + \delta_z E_x) & (\delta_y E_z + \delta_z E_y) & 2\delta_z E_z + div(\vec{E}) \end{bmatrix} =$$
$$(2S_e(p) + tr(S_e(p))I) - K_t(p)(2S(p) + tr(S(p))I) \tag{B.17}$$

Bibliography

Ahmed, M Masroor and Mohamad, Dzulkifli Bin, 2008. Segmentation of brain mr images for tumor extraction by combining kmeans clustering and perona-malik anisotropic diffusion model. *International Journal of Image Processing*, 2(1):27–34.

Akenine-Möller, Tomas, Haines, Eric, Hoffman, Naty, Pesce, Angelo, Iwanicki, Michał and Hillaire, Sébastien, 2018. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press.

Ashikhmin, Michael and Shirley, Peter, 2000. An anisotropic phong brdf model. *J. Graph. Tools*, 5(2):25–32.

Barla, Pascal, Thollot, Joëlle and Markosian, Lee, 2006. X-toon: An extended toon shader. In Douglas DeCarlo and Lee Markosian, editors, *International Symposium on Non-Photorealistic Animation and Rendering (NPAR'06)*. ACM.

Bavoil, Louis and Myers, Kevin, 2008. Order independent transparency with dual depth peeling. Technical report, NVIDIA.

Bavoil, Louis, Sainz, Miguel and Dimitrov, Rouslan, 2008. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks*, SIGGRAPH '08, pages 22:1–22:1. ACM.

Blinn, James F., 1977. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198.

Brown, Pat, Koch, Daniel and Kessenich, John, 2012. Compute shader. URL https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_compute_shader.txt

Bruckner, Stefan and Gröller, Meister Eduard, 2007. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724. Eurographics 2007 3rd Best Paper Award.

Burley, Brent and Walt Disney Animation, Studios, 2012. Physically-based shading at disney. In *ACM SIGGRAPH*, pages 1–7.

Callahan, Steven Paul, 2005. *The k-buffer and its applications to volume rendering*. PhD Thesis, Citeseer.

Carminati, Rémi, 2016. Ondes en milieux complexes. ESPCI.

Carnecky, Robert, Fuchs, Raphael, Mehl, Stephanie, Jang, Yun and Peikert, Ronald, 2013. Smart transparency for illustrative visualization of complex flow surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5):838–851.

Carpenter, Loren, 1984. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 103–108. ACM.

Case, Kenneth Myron and Zweifel, Paul Frederick, 1967. Addison-Wiley.

Cerezo, Eva, Pérez, Frederic, Pueyo, Xavier, Seron, Francisco J. and Sillion, François X., 2005. A survey on participating media rendering techniques. *Vis. Comput.*, 21(5):303–328.

Chandrasekhar, S., 1950. *Radiative Transfer*. Dover books on physics and engineering. Clarendon Press.

Cole, Forrester, Golovinskiy, Aleksey, Limpaecher, Alex, Barros, Heather Stoddart, Finkelstein, Adam, Funkhouser, Thomas and Rusinkiewicz, Szymon, 2008. Where do people draw lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3).

Comaniciu, Dorin, Engel, Klaus, Georgescu, Bogdan and Mansi, Tommaso, 2016. Shaping the future through innovations: From medical imaging to precision medicine. *CoRR*, abs/1605.02029.

Cook, R. L. and Torrance, K. E., 1982. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24.

DeCarlo, Doug, Finkelstein, Adam, Rusinkiewicz, Szymon and Santella, Anthony, 2003. Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):848–855.

DeCarlo, Doug and Rusinkiewicz, Szymon, 2007. Highlight lines for conveying shape. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*.

Donnelly, William and Lauritzen, Andrew, 2006. Variance shadow maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 161–165. ACM.

Dufay, Arthur, 2017. *High quality adaptive rendering of complex photometry virtual environments*. Theses, Université de Bordeaux.

Dutre, Philip, Bala, Kavita, Bekaert, Philippe and Shirley, Peter, 2006. *Advanced Global Illumination*. AK Peters Ltd.

EISEMANN, Elmar, SCHWARZ, Michael, ASSARSSON, Ulf and WIMMER, Michael, 2011. *Real-Time Shadows*. A. K. Peters, Ltd., 1st edition.

ENGEL, Klaus, 2016. Real-time monte-carlo path tracing of medical volume data. URL http://on-demand.gputechconf.com/gtc/2016/presentation/s6535-klaus-engel-real-time-monte-carlo-path-tracing-medical-volume-data.pdf

EVERITT, Cass, 2001. Interactive order-independent transparency. *White paper, nVIDIA*, 2(6):7.

FICK, Adolph, 1855. V. on liquid diffusion. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 10(63):30–39.

FLEMING, Roland W, TORRALBA, Antonio and ADELSON, Edward H, 2004. Specular reflections and the perception of shape. *Journal of Vision*, 4(9):10.

GOOCH, Amy, GOOCH, Bruce, SHIRLEY, Peter and COHEN, Elaine, 1998. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452. ACM.

GOOCH, Bruce and GOOCH, Amy, 2001. *Non-Photorealistic Rendering*. A. K. Peters, Ltd.

GORAL, Cindy M., TORRANCE, Kenneth E., GREENBERG, Donald P. and BATTAILE, Bennett, 1984. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222.

GRANIER, Xavier, VERGNE, Romain, PACANOWSKI, Romain, BARLA, Pascal and REUTER, Patrick, 2012. Enhancing surface features with the Radiance Scaling Meshlab Plugin. In Angeliki Chrysanthi, David Wheatley, Iza Romanowska, Constantinos Papadopoulos, Patricia Murrieta-Flores, Tim Sly, Graeme Earl and Philip Verhagen, editors, *Computer Applications and Quantitative Methods in Archaeology (CAA) 2012*, tome 2 of *Archaeology in the Digital Era*, pages 417–421. Amsterdam University Press.

GRAY, Jim and REUTER, Andreas, 1992. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 1st edition.

GÜNTHER, Tobias, SCHULZE, Maik, ESTURO, Janick Martinez, RÖSSL, Christian and THEISEL, Holger, 2014. Opacity optimization for surfaces. In *Computer Graphics Forum*, tome 33, pages 11–20. Wiley Online Library.

HABLE, John, 2010. URL http://filmicworlds.com/blog/filmic-tonemapping-operators/

HACHISUKA, Toshiya, OGAKI, Shinji and JENSEN, Henrik Wann, 2008. Progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, tome 27, page 130. ACM.

Hadwiger, Markus, Kniss, Joe M., Rezk-salama, Christof, Weiskopf, Daniel and Engel, Klaus, 2006. *Real-time Volume Graphics*. A. K. Peters, Ltd.

Hadwiger, Markus, Ljung, Patric, Salama, Christof Rezk and Ropinski, Timo, 2009. Advanced illumination techniques for gpu-based volume raycasting. In *ACM SIGGRAPH 2009 Courses*, SIGGRAPH '09, pages 2:1–2:166. ACM.

Hadwiger, Markus, Sigg, Christian, Scharsach, Henning, Bühler, Khatja and Gross, Markus, 2005. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 24(3):303–312.

Haskell, Richard C., Svaasand, Lars O., Tsay, Tsong-Tseh, Feng, Ti-Chen, McAdams, Matthew S. and Tromberg, Bruce J., 1994. Boundary conditions for the diffusion equation in radiative transfer. *J. Opt. Soc. Am. A*, 11(10):2727–2741.

Hecht, Eugene, 2002. *Optics*. Pearson Addison Wesley.

Hummel, Mathias, Garth, Christoph, Hamann, Bernd, Hagen, Hans and Joy, Kenneth I, 2010. Iris: Illustrative rendering for integral surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1319–1328.

Interrante, Victoria, Fuchs, Henry and Pizer, Stephen, 1995. Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization'95*, page 52. IEEE Computer Society.

Inventor, Open, . Open inventor toolkit.
URL https://www.openinventor.com/

Ishimaru, A., 1997. *Wave Propagation and Scattering in Random Media*. An IEEE OUP classic reissue. IEEE Press. ISBN 9780780334090.

Jarabo, Adrian, Aliaga, Carlos and Gutierrez, Diego, 2018. A radiative transfer framework for spatially-correlated materials. *ACM Trans. Graph.*, 37(4):83:1–83:13.

Jensen, Henrik Wann, 1996. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 21–30. Springer-Verlag.

Jensen, Henrik Wann and Christensen, Per H., 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 311–320. ACM.

Jensen, Henrik Wann, Marschner, Stephen R., Levoy, Marc and Hanrahan, Pat, 2001. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 511–518. ACM.

Judd, Tilke, Durand, Frédo and Adelson, Edward, 2007. Apparent ridges for line drawing. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07. ACM.

Jönsson, D., Kronander, J., Ropinski, T. and Ynnerman, A., 2012. Historygrams: Enabling interactive global illumination in direct volume rendering using photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2364–2371.

Jönsson, Daniel, Sundén, Erik, Ynnerman, Anders and Ropinski, Timo, 2013. A survey of volumetric illumination techniques for interactive volume rendering. *Computer Graphics Forum*, 33(1):27–51.

Kajiya, James T. and Von Herzen, Brian P, 1984. Ray tracing volume densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 165–174. ACM.

Kautz, Jan, Vázquez, Pere-Pau, Heidrich, Wolfgang and Seidel, Hans-Peter, 2000. *A unified approach to prefiltered environment maps*. Springer.

Khvolson, Orest Danilovich, 1890. *Grundzüge einer mathematischen theorie der inneren diffusion des lichtes*, tome 33.

Kindlmann, Gordon, Whitaker, Ross, Tasdizen, Tolga and Möller, Torsten, 2003. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Visualization, 2003. VIS 2003. IEEE*, pages 513–520. IEEE.

Kirchhoff, G. and Bunsen, R., 1860. Chemische analyse durch spectralbeobachtungen. *Annalen der Physik*, 186(6):161–189.

Koerner, David, Portsmouth, Jamie and Jakob, Wenzel, 2018. PN-Method for Multiple Scattering in Participating Media. In Wenzel Jakob and Toshiya Hachisuka, editors, *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association.

Kolomenkin, Michael, Shimshoni, Ilan and Tal, Ayellet, 2008. Demarcating curves for shape illustration. In *ACM Transactions on Graphics (TOG)*, tome 27, page 157. ACM.

Krissian, Karl, 2002. Flux-based anisotropic diffusion applied to enhancement of 3-d angiogram. *IEEE transactions on medical imaging*, 21(11):1440–1442.

Kronander, J., Jonsson, D., Low, J., Ljung, P., Ynnerman, A. and Unger, J., 2012. Efficient visibility encoding for dynamic illumination in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):447–462.

Kubisch, Christoph, 2014. Order independent transparency in opengl 4.x. In *GPU Technology Conference (GTC)*.

L Sandell, Julia and Zhu, Timothy, 2011. A review of in-vivo optical properties of human tissues and its impact on pdt. 4:773–87.

Lagarde, Sébastien, 2012. Amd cubemapgen for physically based rendering. URL https://seblagarde.wordpress.com/2012/06/10/amd-cubemapgen-for-physically-based-rendering/

Lansdown, J. and Schofield, S., 1995. Expressive rendering: a review of nonphotorealistic techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37.

Lewis, Robert R., 1993. Making shaders more physically plausible. Technical report.

Ljung, Patric, Krüger, Jens, Gröller, Eduard, Hadwiger, Markus, Hansen, Charles D. and Ynnerman, Anders, 2016. State of the art in transfer functions for direct volume rendering. In *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: State of the Art Reports*, EuroVis '16, pages 669–691. Eurographics Association.

Loos, Bradford James and Sloan, Peter-Pike, 2010. Volumetric obscurance. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 151–156. ACM.

Mahesh, Mahadevappa, 2011. Fundamentals of medical imaging. *Medical Physics*, 38(3):1735–1735.

Maxima, . Maxima. URL http://maxima.sourceforge.net/

McGuire, Morgan and Mara, Michael, 2016. A phenomenological scattering model for order-independent transparency. In *I3D 2016*, page 10.

McGuire, Morgan, Osman, Brian, Bukowski, Michael and Hennessy, Padraic, 2011. The alchemy screen-space ambient obscurance algorithm. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 25–32. ACM.

Montes, Rosana and Ureña, Carlos, 2012. An overview of brdf models.

Morar, Anca, Moldoveanu, Florica and Gröller, Eduard, 2012. Image segmentation based on active contours without edges. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, pages 213–220. IEEE.

Murray, David, Baril, Jérôme and Granier, Xavier, 2016. Shape Depiction for Transparent Objects with Bucketed k-Buffer. In *EuroGraphics Symposium on Rendering*.

Nicodemus, F. E., 1970. Reflectance nomenclature and directional reflectance and emissivity. *Appl. Opt.*, 9:1474–1475.

Ohtake, Yutaka, Belyaev, Alexander and Seidel, Hans-Peter, 2004. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics (TOG)*, 23(3):609–612.

PERONA, P. and MALIK, J., 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639.

PHONG, Bui Tuong, 1975. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317.

PIERRAT, Romain, 2007. *Propagation et émission du rayonnement en milieu diffusant. Application à l'imagerie des milieux complexes.* PhD Thesis, École Centrale Paris.

PLANCK, M. and MASIUS, M., 1914. *The Theory of Heat Radiation.*

PORTER, Thomas and DUFF, Tom, 1984. Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 253–259. ACM.

RAMAMOORTHI, Ravi and HANRAHAN, Pat, 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500. ACM.

RAUTEK, Peter, BRUCKNER, Stefan and GROLLER, Eduard, 2007. Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1336–1343.

REINHARD, Erik, WARD, Greg, PATTANAIK, Sumanta and DEBEVEC, Paul, 2005. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc.

RITSCHEL, Tobias, 2007. Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In Paolo Cignoni and Jiri Sochor, editors, *EG Short Papers*. The Eurographics Association.

RITSCHEL, Tobias, DACHSBACHER, Carsten, GROSCH, Thorsten and KAUTZ, Jan, 2012. The state of the art in interactive global illumination. *Comput. Graph. Forum*, 31(1):160–188.

SAITO, Takafumi and TAKAHASHI, Tokiichiro, 1990. Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 197–206. ACM.

SCHENKE, Stefan, WUENSCHE, Burkhard C and DENZLER, Joachim, 2005. Gpu-based volume segmentation. In *Proceedings of IVCNZ*, tome 5, pages 171–176.

SCHUSTER, Arthur, 1905. Radiation through a foggy atmosphere. 21.

SHADERTOY, . Shadertoy.
URL https://www.shadertoy.com/

Shi, Lin, Liu, Wen, Zhang, Heye, Xie, Yongming and Wang, Defeng, 2012. A survey of gpu-based medical image computing techniques. *Quantitative imaging in medicine and surgery*, 2(3):188.

Sillion, Francois and Puech, Claude, 1994. *Radiosity and Global Illumination*. Morgan Kaufmann.

Sloan, Peter-Pike J, Martin, William, Gooch, Amy and Gooch, Bruce, 2001. The lit sphere: A model for capturing npr shading from art. In *Graphics interface*, tome 2001, pages 143–150. Citeseer.

Stam, Jos, 1995. Multiple scattering as a diffusion process. In Patrick M. Hanrahan and Werner Purgathofer, editors, *Rendering Techniques '95*, pages 41–50. Springer Vienna. ISBN 978-3-7091-9430-0.

Strauss, Paul S., 1993. Iris inventor, a 3d graphics toolkit. In *Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '93, pages 192–200. ACM, New York, NY, USA. ISBN 0-89791-587-9. doi:10.1145/165854.165889.
URL http://doi.acm.org/10.1145/165854.165889

Strothotte, Thomas and Schlechtweg, Stefan, 2002. *Non-photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann Publishers Inc.

Tatarchuk, Natalya, Shopf, Jeremy and DeCoro, Christopher, 2007. Real-time isosurface extraction using the gpu programmable geometry pipeline. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, pages 122–137. ACM.

Thiedemann, Sinje, Henrich, Niklas, Grosch, Thorsten and Müller, Stefan, 2011. Voxel-based global illumination. In *Symposium on Interactive 3D Graphics and Games*, pages 103–110. ACM.

van de Hulst, H.C., 1981. Light scattering by small particles. 10.

Vardis, Kostas, Vasilakis, Andreas A. and Papaioannou, Georgios, 2016. A multiview and multilayer approach for interactive ray tracing. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, pages 171–178. ACM.

Veach, Eric and Guibas, Leonidas J., 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76. ACM Press/Addison-Wesley Publishing Co.

Vergne, Romain, Barla, Pascal, Granier, Xavier and Schlick, Christophe, 2008. Apparent relief: a shape descriptor for stylized shading. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 23–29.

VERGNE, Romain, PACANOWSKI, Romain, BARLA, Pascal, GRANIER, Xavier and SCHLICK, Christophe, 2009. Light warping for enhanced surface depiction. *ACM Transaction on Graphics (Proceedings of SIGGRAPH 2009)*, 28(3).

VERGNE, Romain, PACANOWSKI, Romain, BARLA, Pascal, GRANIER, Xavier and SCHLICK, Christophe, 2010. Radiance scaling for versatile surface enhancement. In *I3D '10: Proc. symposium on Interactive 3D graphics and games*. ACM.

WERNECKE, Josie, 1994a. *The Inventor Mentor : Programming Object-oriented 3D graphics with Open Inventor, Release 2*, tome 1.

WERNECKE, Josie, 1994b. *The Inventor Toolemaker : Extending Open Inventor, Release 2*, tome 1.

WHITTED, Turner, 1980. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349.

WIEN, Willy, 1894. Temperatur und entropie der strahlung. *Annalen der Physik*, 288(5):132–165.

WILLIAMS, Lance, 1978. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274.

WOO, Andrew and POULIN, Pierre, 2012. *Shadow Algorithms Data Miner*.

YANG, Jason C, HENSLEY, Justin, GRÜN, Holger and THIBIEROZ, Nicolas, 2010. Real-time concurrent linked list construction on the gpu. In *Computer Graphics Forum*, tome 29, pages 1297–1304. Wiley Online Library.

ZHANG, Yubo and MA, Kwan-Liu, 2013. Fast global illumination for interactive volume visualization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pages 55–62. ACM.