Discrete shape analysis for global illumination

Laurent Noel

UNIVERSITÉ PARIS-EST

PARIS-EST UNIVERSITY

DOCTORAL SCHOOL MSTIC

A thesis submitted in partial fulfillment for the degree of *Doctor of Philosophy* in Computer Science

presented by

LAURENT NOËL

advised by VENCESLAS BIRI

DISCRETE SHAPE ANALYSIS FOR GLOBAL ILLUMINATION

December 2015

Comitee in charge:

DANIEL MENEVEAUX (reviewer)
CHRISTOPHE RENAUD (reviewer)
CHRISTOPHE LOHOU
LAURENT LUCAS
SYLVAIN LEFEBVRE
VENCESLAS BIRI

ABSTRACT

Nowadays, computer generated images can be found everywhere, through a wide range of applications such as video games, cinema, architecture, publicity, artistic design, virtual reality, scientific visualization, lighting engineering, etc. Consequently, the need for visual realism and fast rendering is increasingly growing. Realistic rendering involves the estimation of global illumination through light transport simulation, a time consuming process for which the convergence rate generally decreases as the complexity of the input virtual 3D scene increases. In particular, occlusions and strong indirect illumination are global features of the scene that are difficult to handle efficiently with existing techniques. This thesis addresses this problem through the application of discrete shape analysis to rendering.

Our main tool is a curvilinear skeleton of the empty space of the scene, a sparse graph containing important geometric and topological information about the structure of the scene. By taking advantage of this skeleton, we propose new methods to improve both real-time and off-line rendering methods. Concerning real-time rendering, we exploit geometric information carried by the skeleton for the approximation of shadows casted by a large set of virtual point lights representing the indirect illumination of the 3D scene. Regarding off-line rendering, our works focus on algorithms based on path sampling, that constitute the main paradigm of state-of-the-art methods addressing physically based rendering. Our skeleton leads to new efficient path sampling strategies guided by topological and geometric features. Addressing the same problem, we also propose a sampling strategy based on a second tool from discrete shape analysis: the opening function of the empty space of the scene, describing the local thickness of that space at each point.

Our contributions demonstrate improvements over existing approaches and clearly indicate that discrete shape analysis offers many opportunities for the development of new rendering techniques.

KEYWORDS computer graphics, realistic rendering, global illumination, path tracing, discrete shape analysis, skeletonization

RÉSUMÉ

Les images de synthèse sont présentes à travers un grand nombre d'applications tel que les jeux vidéo, le cinéma, l'architecture, la publicité, l'art, la réalité virtuelle, la visualisation scientifique, l'ingénierie en éclairage, etc. En conséquence, la demande en photoréalisme et techniques de rendu rapide ne cesse d'augmenter. Le rendu réaliste d'une scène virtuelle nécessite l'estimation de son illumination globale grâce à une simulation du transport de lumière, un processus coûteux en temps de calcul dont la vitesse de convergence diminue généralement lorsque la complexité de la scène augmente. En particulier, une forte illumination indirecte combinée à de nombreuses occlusions constitue une caractéristique globale de la scène que les techniques existantes ont du mal à gérer. Cette thèse s'intéresse à ce problème à travers l'application de techniques d'analyse de formes pour le rendu 3D.

Notre principal outil est un squelette curviligne du vide de la scène, représenté par un graphe contenant des informations sur la topologie et la géométrie de la scène. Ce squelette nous permet de proposer de nouvelles méthodes pour améliorer des techniques de rendu temps réel et non temps réel. Concernant le rendu temps réel, nous utilisons les informations géométriques du squelette afin d'approximer le rendu des ombres projetés par un grand nombre de points virtuels de lumière représentant l'illumination indirecte de la scène 3D. Pour ce qui est du rendu non temps réel, nos travaux se concentrent sur des algorithmes basés sur l'échantillonnage de chemins, constituant actuellement le principal paradigme en rendu physiquement plausible. Notre squelette mène au développement de nouvelles stratégies d'échantillonnage de chemins, guidés par des caractéristiques topologiques et géométriques. Nous adressons également ce problème à l'aide d'un second outil d'analyse de formes: la fonction d'ouverture du vide de la scène, décrivant l'épaisseur locale du vide en chacun de ses points.

Nos contributions offrent une amélioration des méthodes existantes and indiquent clairement que l'analyse de formes offre de nombreuses opportunités pour le développement de nouvelles techniques de rendu 3D.

MOTS CLEFS synthèse d'images, rendu réaliste, illumination globale, lancer de rayons, analyse de formes discrètes, squelettisation

PUBLICATIONS

[CNBC13] John Chaussard, Laurent Noël, Venceslas Biri, and Michel Couprie. A 3d curvilinear skeletonization algorithm with application to path tracing. In *Discrete Geometry for Computer Imagery*, volume 7749 of *Lecture Notes in Computer Science*, pages 119–130. Springer Berlin Heidelberg, 2013.

[NB14a] Laurent Noël and Venceslas Biri. Real-time global illumination for games using topological information. In *Annual International Conference on Computer Games Multimedia and Allied Technology*, Lecture Notes in Computer Science, 2014.

[NB14b] Laurent Noël and Venceslas Biri. Real-time global illumination using topological information. In *GSTF Journal on Computing*, 2014.

[NB15a] Laurent Noël and Venceslas Biri. Portal extraction based on an opening labeling for ray tracing. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, volume 9082 of *Lecture Notes in Computer Science*, pages 27–38. Springer International Publishing, 2015.

[NB15b] Laurent Noël and Venceslas Biri. Skeleton based vertex connection resampling for bidirectional path tracing. In *23rd Pacific Conference on Computer Graphics and Applications*, Pacific Graphics, 2015.

[NCB12] Laurent Noël, John Chaussard, and Venceslas Biri. Coarse irradiance estimation using curvilinear skeleton. In *ACM SIGGRAPH 2012 Posters*, SIGGRAPH '12, pages 101:1–101:1, New York, NY, USA, 2012. ACM.

REMERCIEMENTS

Je tiens tout d'abord à remercier Venceslas Biri, mon directeur de thèse, pour m'avoir offert l'opportunité de travailler en thèse sur un domaine qui me passionne, la synthèse d'images. Il a su me supporter pendant 3 ans, critiquer mes idées parfois (souvent ?) mauvaises, me remettre sur le droit chemin, être mon MJ sur plusieurs sessions de jeux de rôle, et me laisser la liberté de gérer mon emploi du temps et mon travail. Un grand merci également aux autres personnes constituant mon jury de thèse, Daniel Meneveaux, Christophe Renaud, Christophe Lohou, Laurent Lucas et Sylvain Lefebvre, pour le temps consacré à la relecture et à l'évaluation de mes travaux.

Merci aux doctorants de l'équipe A3SI, qui seront bientôt dans la même galère que moi (l'écriture de la thèse !), Odyssée, Elodie, Eloïse, Clara, Julie, Ali et Thibault, et aux ingénieurs de recherche (la plupart ayant déserté à l'heure où j'écris ces mots), Mathieu, Geoffrey et Diane, pour tous les bons moments passés au labo, les heures perdues à jouer au loup-garou, les jeux de rôle et les jeux de plateau le soir, qui aident vraiment à se détendre et à tisser des liens :) Merci également aux deux doctorants de ma promotion de master, Gregory et Jérôme, qui ont malheureusement choisi un domaine de recherche impur, mais qui ont le mérite de m'avoir fait découvert le Saigon, un bon restaurant Vietnamien, pas trop cher donc accessible aux faibles revenus d'un doctorant :p

Je remercie les deux étudiants qui ont fait leur stage de master 1 au labo, et qui m'ont assisté dans mes travaux, Thomas Demenat, sans qui je n'aurais jamais eu une implémentation de l'algorithme de voxelisation/squelettisation dans mon moteur de rendu, et Jefferson Mangue, qui a codé la majeure partie de l'algorithme de shadow mapping basé sur le squelette.

Merci à l'ensemble de l'équipe A3SI, au sein de laquelle j'ai été intégré durant ma thèse. Je souhaite remercier Michel Couprie et Gilles Bertrand, qui ont répondu à mes questions concernant la topologie discrète (certaines de ces questions n'ayant sans doute aucun sens). Merci à Hugues Talbot et Laurent Najman, avec qui les discussions autour de la morphologie mathématique et l'optimisation ont permis de faire germer quelques idées dans ma pauvre tête. Merci à John Chaussard, pour avoir développé l'algorithme de squelettisation utilisé pour la plupart des travaux présentés dans ce manuscrit, ainsi que pour sa pédagogie dans l'explication des concepts relatifs aux cubiques complexes. Une mention spéciale également pour ses vannes, qui n'ont d'égal que certaines blagues, souvent moyennes, de Vincent Nozick. Merci à ce dernier pour son support et ses conseils durant l'ensemble de ma thèse, pour son fameux "contrôle du travail", et ses blagues sur mes pieds :) Merci également à Benjamin Raynal, pour ses passages dans mon bureau afin de me rappeler régulièrement le peu de temps qu'il me restait pour finir mon manuscrit :p

Je voudrais également remercier l'ensemble des personnes du LIGM, qui ont été mes professeurs au cours de mes études universitaires. C'est principalement grâce à eux que j'ai décidé de continuer en thèse (une folie... principalement justifiée par le barbecue annuel organisé par le labo !).

Je souhaite également remercier mes étudiants de l'école d'ingénieurs Imac, que j'ai eu en TD, pour m'avoir laissé participer à leurs-week ends d'intégration et m'avoir supporté comme enseignant :p

Un grand merci à toutes les personnes qui ne sont pas liés à l'université mais qui qui m'ont apporté un énorme soutien au cours de ces années. Merci a mes amis, Kévin, qui a continué en thèse à Rennes (lacheur ! mais Rennes c'est pas mal comme ville, donc je te pardonne), Bryan et Vina, qui m'aiment tellement qu'ils ont décidé de venir vivre près de chez moi pour quelques mois avant d'emmenager dans leur maison, Vincent, que j'ai rejoint chez Ubisoft à la fin de mon contrat de thèse afin de lui rappeler qui est le patron :p, et Jordan, qui a apprecié de squatter mon appartement au cours d'un été pour être plus proche de son stage (et j'ai moi même particulièrement apprecié qu'il m'emmene en voiture au labo le matin). Merci à ma soeur Céline, ma grand mère Marie, et ma tante Christine, pour leur soutien, les bons moments passés à leur coté et les repas du week end qui aident à relacher la pression. Merci à mon frère Julien, qui habite loin mais qui à eu l'occasion de passer quelques semaines chez moi au cours de ma thèse. J'espère qu'il se rapprochera de la région parisienne plus tard afin qu'on puisse passer plus de temps en famille. Un merci spécial à ma moitié, Atalis, qui a su m'aimer, me supporter, et rendre les moments difficiles plus supportables. Merci à toi d'avoir accepté mes absences au cours des derniers mois, consacrés à l'écriture du manuscrit et à la finition des derniers articles, et désolé de ne pas avoir pu t'accompagner en vacance, on ratrappera ça ! Un grand merci également a ses parents, Carmina et François, et sa soeur Karina, pour m'avoir accueilli la plupart des week ends, et plusieurs semaines de vacances en Espagne !

CONTENTS

1

INTRODUCTION

Nowadays, computer generated images can be seen everywhere, both in entertainment applications (video games, cinema, VFX) and professional applications (lighting engineering, scientific visualization, computer aided design, digital art, product design). This can be explained by the strong ability of images to convey information, compared to other forms of medias such as text or sound. As the adage goes,"A picture is worth a thousand words".

1.1 REALISTIC RENDERING

One of the most challenging problem faced by computer graphics is realistic 3D rendering, aiming at a precise simulation of light transport in a virtual scene to produce photo-realistic images or accurate measurements. Demands for realistic rendering have increased over the past decades, even for applications that require real-time rendering such as video games or virtual reality simulators. This has been made possible thanks to new hardware like GPUs or many-core CPUs, that deliver the necessary power to perform massive parallel computations, that are common in rendering. However, even with more computational power, light transport simulation implies mathematical challenges that need to be solved algorithmically in order to produce photo-realistic images in a decent amount of time. Light transport simulation involves the costly computation of global illumination, that models all possible interactions of light with the virtual scene that is to be rendered. Most of current algorithmic approaches attack this problem with stochastic sampling, to explore the space of light paths that connect light sources with the camera, and Monte Carlo integration, to estimate the intensity of each pixel. The challenge faced by these methods is to deliver a solution that works well for any kind of scenes. Recent works have mainly be focused on solving the problem of rendering scenes containing heterogeneous mixtures of rich materials, that are now common in industry. Moreover, the complexity of a scene can also come from its geometry and topology. When many occlusions are present, stochastic sampling of contributing paths can become inefficient, if the sampling strategy does not take into account global information about visibility. This thesis explores new solutions to take advantage of such information, thanks to techniques from the framework of discrete shape analysis.

1.2 DISCRETE SHAPE ANALYSIS FOR RENDERING

This thesis is focused on the rendering of complex scenes exposing many occlusions. Such configurations are generally difficult to render efficiently because occlusions produce discontinuities in the distribution of lighting energy in the scene. Extracting usable information

from the scene geometry, in order to adapt rendering algorithms to the visibility configuration, can be hard, given that input scenes are generally represented by soups of triangles. Most of existing approaches only use local information at each point to compute its illumination, ignoring possible occlusions around the point.

The major trend in research on global illumination is Monte Carlo estimation based on stochastic local path sampling. In this framework, a path carrying light energy is obtained by connecting two random sub-path, one starting at a light source and the other starting at the camera. For Monte Carlo estimation to be efficient, paths having a high contribution on the image must be sampled more frequently. To sample a sub-path, the main procedure involved is *ray sampling*, that selects a random outgoing direction at the last vertex of a sub-path in order to extend it. Ideally, rays must be sampled to reach important parts of the scene more often (bright parts of the scene or parts visible from the camera). The second procedure for path sampling is *vertex connection*, performing a visibility test between the last vertices of two different sub-paths in order to construct a complete path connecting a light source with the camera. These two procedures can become highly inefficient in complex scenes since occlusions are unpredictable from a local sampling point of view.

Discrete shape analysis (DSA) offers many tools to extract geometric or topological information from a volumetric 3D object, such as the empty space of a scene, the medium where light travels. In this thesis, we propose new methods to take advantage of such tools in order to improve existing rendering algorithms. We focused our work on two main tools from the DSA framework: topological skeletons (from digital topology) and opening maps (from mathematical morphology). Our idea of using a topological skeleton comes from the fact that visibility in a scene is related to the topological structure of its empty space. Each surface creates holes or tunnels in the empty space, forcing the light to travel through more complex paths. The extraction of a skeleton of the empty space of the scene offers a sparse representation of its geometry and topology, that can be beneficial to rendering algorithms. An opening map of a digital object describes the local thickness of this object at each point. This information is valuable for rendering algorithm since narrow regions of the empty space are usually hard to explore without any prior knowledge. Given these tools, we address the following questions in this thesis:

— How can we use a topological skeleton to make rendering algorithms based on path sampling more robust in occluded scenes ?
— Is it possible to take advantage of the same kind of skeleton to also improve real-time rendering of global illumination ?
— How to take advantage of the information stored by an opening map to guide path sampling on regions of empty space that are usually hard to explore ?

To provide answers to these questions, this thesis offers new rendering methods, both for off-line and interactive/real-time rendering, that are inscribed in the following themes:

SKELETON FOR MONTE CARLO RENDERING ALGORITHMS. We introduce new methods to exploit a curvilinear skeleton of the empty space of the scene to improve path sampling for Monte Carlo rendering algorithms such as path tracing and bidirectional path tracing. This skeleton is used to improve both ray sampling and vertex connection, the two key components of path sampling.

SKELETON FOR INTERACTIVE MANY-LIGHT RENDERING. Many-light rendering approximates the illumination of a scene with a high number of virtual point lights, involving a high cost in the evaluation of shadows. We propose two new methods to approximate these shadows in the context of interactive/real-time rendering, using segmentations extracted from a curvilinear skeleton of the empty space of the scene.

OPENING MAPS FOR RAY SAMPLING. Using an opening map of the empty space, we are able to partition it in regions of variable thickness. With this partitioning, we can extract portals separating regions to efficiently sample rays from one region to the other, independently from their relative thickness.

1.3 THESIS ORGANIZATION

This thesis is divided in 12 chapters, distributed across 2 parts.

Light transport simulation is presented in Part I, comprising three chapters. Chapter 2 presents the mathematical and physical background forming the basis of light transport simulation. Monte Carlo integration, the stochastic estimation strategy favored by most rendering algorithms nowadays, is exposed in Chapter 3. Chapter 4 gives an overview of the main rendering solutions based on path sampling, from which most state-of-the-art methods are derived.

The Part II of this thesis is dedicated to the use of our tools from DSA to the global illumination problem. We give in Chapter 5 an overview of thinning algorithms, which are used to compute discrete skeletons. Chapter 6 details the thinning algorithm we developed to compute a topological curvilinear skeleton of the empty space of the scene.

Chapter 7 and Chapter 8 both present new methods to approximate, in real-time, the rendering of indirect illumination. The former proposes an approach based on the concept of visibility gates extracted from the skeleton, while the latter presents an approach based on shadow mapping performed at skeleton points, providing an efficient approximation of shadows casted from many virtual light sources.

Subsequent chapters are dedicated to sampling strategies for realistic off-line rendering. The problem of ray sampling guided by shape analysis is addressed by Chapter 9 and Chapter 10. Chapter 9 proposes a method based on the skeleton topology and geometry to sample rays in preferred directions that lead to important parts of the scene. We use an opening map of the empty space for the same purpose, in the sampling strategy presented in Chapter 10. This opening map helps us for the extraction portals of the empty space that separate thick regions from narrow ones, such that rays can be guided through these portals efficiently. While this work is still in progress, we present first rendering results and discuss problems that have to be solved in order to obtain a robust rendering solution. The problem of vertex connection for path sampling is addressed by our resampling strategy presented in Chapter 11. For that, we use a segmentation of the scene obtained from our skeleton, and drive the resampling based on visibility and geometric information extracted from this segmentation.

Finally, Chapter 12 concludes the works presented in this thesis and discusses perspectives for future works regarding the application of discrete shape analysis to rendering problems.

Part I

LIGHT TRANSPORT SIMULATION

2

MATHEMATICAL FORMULATION OF LIGHT TRANSPORT

To visually perceive the real world, our eyes are sensitive to incident light by the mean of biological sensors (rods and cones), that perform measurements from incident light energy. These measurements are then interpreted by our brain as an image of what stands in front of us. Electronic cameras work similarly, with a set of sensors to measure incident light, followed by a conversion to a digital image. Since light plays a major role for the sense of sight, a good understanding of its physical properties is required to develop robust rendering algorithms. Over the history, light has been described by several physical theories, such as the particle theory and the wave theory, before it was recognized that light exhibits the properties of both particles and waves. Visible light is an electromagnetic radiation, with wavelengths varying approximately in the range 380 to 750 nm. Optics is the field of research that studies the propagation of light and its interactions with matter. The most complete physical model is quantum optics, describing lighting phenomena at submicroscopic levels. Yet, the goal of rendering algorithms is to efficiently produce realistic and convincing images, which does not require to simulate all kinds of phenomena. Consequently, realistic rendering is principally based on geometric optics, describing the propagation of light in terms of rays, and radiometry, which assumes that the distribution of light energy in a scene can be completely characterized by a density function. This simple model is able to describe the most important visible phenomena, but is limited since it cannot define diffraction, interference or fluorescence, for example. Nevertheless, the model is sufficient to compute realistic images and accurate measurements in a decent amount of time for applications such as cinema, video games, architecture, lighting engineering or scientific visualization.

In this chapter, we present a mathematical formulation of geometric optics and radiometry. We first introduce, in Section 2.1, geometric quantities involved in light transport simulations, such as the mathematical representation of the 3D scene given as input of a rendering algorithm. Next, we describe radiometric quantities in Section 2.2, defined as functions over various geometric spaces. In Section 2.3, we present a scattering model often used in rendering. We limit our discussion to scattering at surfaces since the works presented in this thesis do not deal with participating media. Finally, we give a description of light transport equations in Section 2.4, that provide practical expressions for measuring incident light energy at a virtual sensor or at any point of the scene.

Contents

2.1 GEOMETRIC QUANTITIES

2.1.1 *Surfaces*

The geometry of the scene is represented by a sequence of $N_{\mathcal{M}}$ surfaces $(\mathcal{M}_i)_{i \in [\![1, N_{\mathcal{M}}]\!]}$ such that each surface $\mathcal{M}_i \subset \mathbb{R}^3$ is a bounded 2-dimensional manifold and the intersection of two surfaces is at most a negligible set for the area measure $\mathcal{A}$:

$$\forall i \neq j, \mathcal{A}(\mathcal{M}_i \cap \mathcal{M}_j) = 0 \tag{1}$$

The scene $\mathcal{M}$ is defined by the union of all surfaces (Figure 1):

$$\mathcal{M} := \bigcup_{i=1}^{N_{\mathcal{M}}} \mathcal{M}_i \tag{2}$$

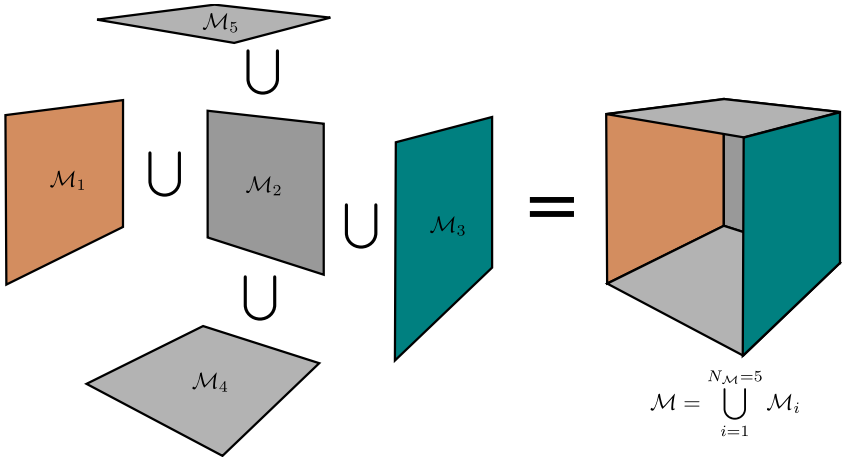


Figure 1 – The scene is defined as the union of individual surfaces.

2.1.2 *Directions*

The sphere of unit vectors, referred as *directions*, is denoted by $\mathcal{S}^2$:

$$\mathcal{S}^2 := \{\omega \in \mathbb{R}^3 \mid \|\omega\| = 1\} \tag{3}$$

The dot product between two directions $\omega_1, \omega_2 \in \mathcal{S}^2$ is noted $\omega_1 \cdot \omega_2 = \cos\theta_{\omega_1,\omega_2}$, where $\theta_{\omega_1,\omega_2}$ is the angle separating the two vectors (Figure 2).

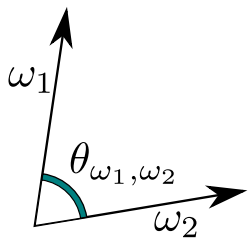


Figure 2 – Illustration of our notation for the angle between two directions.

Each surface point $\mathbf{x} \in \mathcal{M}$ has an associated normal $n_\mathbf{x} \in \mathcal{S}^2$, orthogonal to its tangent plane. We denote by $\mathcal{S}^+_\mathbf{x}$ the positive hemisphere of $\mathbf{x}$ and $\mathcal{S}^-_\mathbf{x}$ its negative hemisphere (Figure 3):

$$\mathcal{S}^+_\mathbf{x} := \{\omega \in \mathcal{S}^2 \mid n_\mathbf{x} \cdot \omega > 0\} \tag{4}$$
$$\mathcal{S}^-_\mathbf{x} := \{\omega \in \mathcal{S}^2 \mid n_\mathbf{x} \cdot \omega < 0\} \tag{5}$$

Given two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$, the unit direction from $\mathbf{x}$ to $\mathbf{y}$ is denoted by $\omega_{\mathbf{x},\mathbf{y}}$ (Figure 3):

$$\omega_{\mathbf{x},\mathbf{y}} := \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|} \tag{6}$$

We also use the notation $\theta_{\mathbf{x},\mathbf{y}}$ for the angle between directions $n_\mathbf{x}$ and $\omega_{\mathbf{x},\mathbf{y}}$, such that $n_\mathbf{x} \cdot \omega_{\mathbf{x},\mathbf{y}} = \cos\theta_{\mathbf{x},\mathbf{y}}$ (Figure 3).

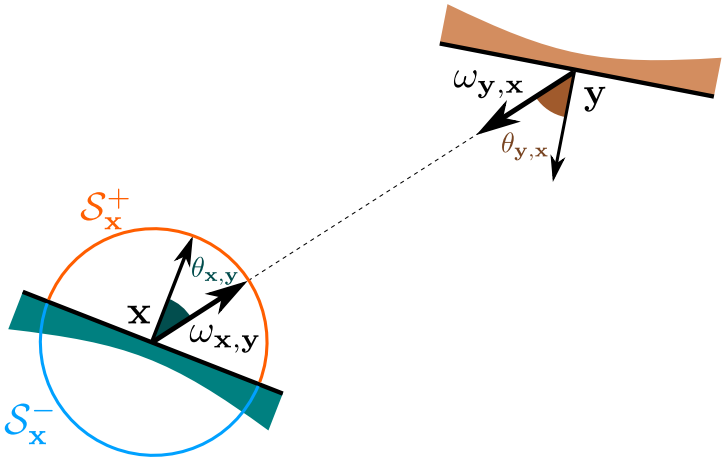


Figure 3 – Illustration of our notation positive and negative hemisphere, as well as directions between two points.

2.1.3 *Ray casting*

A ray $\mathbf{r}$ is a pair $(\mathbf{x}_\mathbf{r}, \omega_\mathbf{r}) \in \mathbb{R}^3 \times \mathcal{S}^2$ where $\mathbf{x}_\mathbf{r}$ is the *origin* of the ray and $\omega_\mathbf{r}$ is its *direction*. The set of rays is noted $\mathcal{R} := \mathbb{R}^3 \times \mathcal{S}^2$ and is referred as the *ray space*.

The set of *surface rays*, having their origin on the scene surfaces, is noted $\mathcal{R}_\mathcal{M} := \mathcal{M} \times \mathcal{S}^2$ and is referred as the *surface ray space*.

The *boundary distance function* [Arv95] $d_\mathcal{M} : \mathcal{R} \to (0, +\infty]$ is defined by (Figure 4):

$$d_\mathcal{M}(\mathbf{x}, \omega) := \inf\{\alpha > 0 \mid \mathbf{x} + \alpha.\omega \in \mathcal{M}\} \tag{7}$$

Let $\mathbf{x} \in \mathbb{R}^3$, we define $\mathbf{x}_\mathcal{M} : \{\omega \in \mathcal{S}^2 \mid d_\mathcal{M}(\mathbf{x}, \omega) < +\infty\} \to \mathcal{M}$, the *ray casting function* at $\mathbf{x}$, by (Figure 4):

$$\mathbf{x}_\mathcal{M}(\omega) := \mathbf{x} + d_\mathcal{M}(\mathbf{x}, \omega).\omega \tag{8}$$

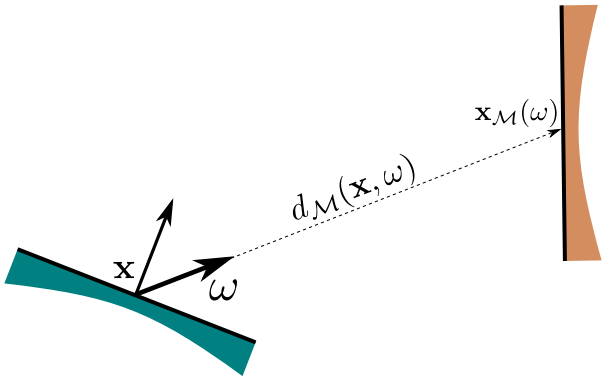


Figure 4 – Boundary distance function and ray casting function.

2.1.4 *Visibility*

The *visibility function* $V : \mathbb{R}^3 \to \{0, 1\}$ is defined by:

$$V(\mathbf{x}, \mathbf{y}) := \begin{cases} 1, \text{ if } ||\mathbf{x} - \mathbf{y}|| \leq d_{\mathcal{M}}(\mathbf{x}, \omega_{\mathbf{x},\mathbf{y}}) \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

2.1.5 *Integration*

Quantities related to light energy are mostly defined by integrals over the ray space. An integral over the ray space can be expressed according to several measures. This section reviews the main integration measures used in rendering.

2.1.5.1 *Solid angle measure*

The *solid angle measure* $\sigma(D)$ of a set of directions $D \subset \mathcal{S}^2$ is defined by the area on the unit sphere covered by D:

$$\sigma(D) = \int_{\omega \in D} d\sigma(\omega) \tag{10}$$

If $\mathcal{S}^2$ is parameterized by the standard spherical mapping $\omega = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, $\theta \in [0, \pi)$, $\phi \in [0, 2\pi)$, then we have (Figure 5):

$$d\sigma(\omega) = |\sin\theta| d\phi d\theta \tag{11}$$

The solid angle of the sphere of directions $\mathcal{S}^2$ is $\sigma(\mathcal{S}^2) = 4\pi$.

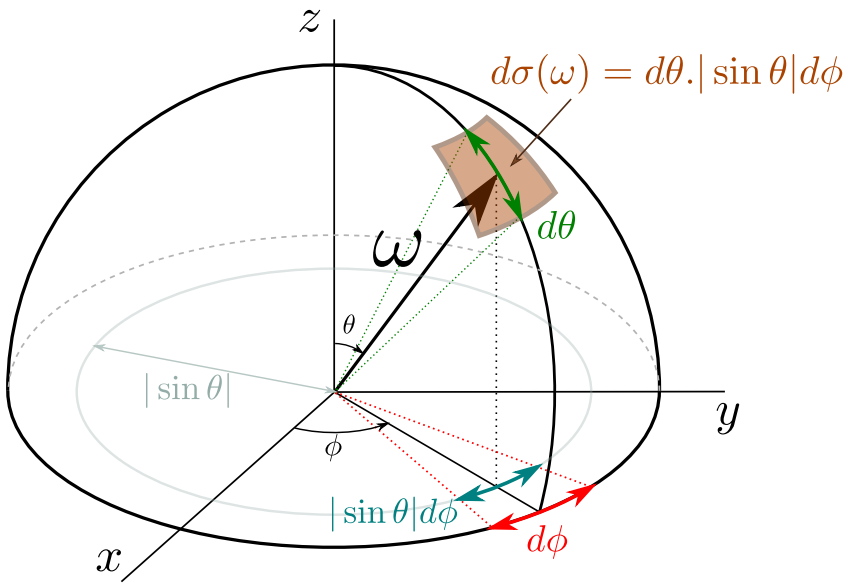


Figure 5 – Illustration of the differential solid angle measure.

2.1.5.2 *Projected solid angle measure*

Let $\mathbf{x} \in \mathcal{M}$ and $f : \mathcal{S}^2 \to \mathbb{R}$ a function defined on directions. A common form of integral involved in light transport simulation is:

$$F = \int_{\omega \in \mathcal{S}^2} f(\omega) |n_{\mathbf{x}} \cdot \omega| d\sigma(\omega) \tag{12}$$

In this expression, $|n_{\mathbf{x}} \cdot \omega| d\sigma(\omega)$ is the differential solid angle around direction ω projected on the tangent plane of $\mathbf{x}$. The *projected solid angle measure* $\sigma^{\perp}_{\mathbf{x}}$ at $\mathbf{x}$ for a set of directions $D \subset \mathcal{S}^2$ is defined by (Figure 6):

$$\begin{aligned} \sigma^{\perp}_{\mathbf{x}}(D) &= \int_{\omega \in D} |n_{\mathbf{x}} \cdot \omega| d\sigma(\omega) \\ &= \int_{\omega \in D} d\sigma^{\perp}_{\mathbf{x}}(\omega) \end{aligned} \tag{13}$$

The integral can then be rewritten more concisely with respect to projected solid angle measure:

$$F = \int_{\omega \in \mathcal{S}^2} f(\omega) d\sigma^{\perp}_{\mathbf{x}}(\omega) \tag{14}$$

2.1.5.3 *Area measure*

Let $\mathbf{x} \in \mathcal{M}$ and $f : \mathcal{R}_{\mathcal{M}} \to \mathbb{R}$ be a function defined on the surface ray space and consider the integral:

$$F = \int_{\omega \in \mathcal{S}^2} f(\mathbf{x}_{\mathcal{M}}(\omega), -\omega) |n_{\mathbf{x}} \cdot \omega| d\sigma(\omega) \tag{15}$$

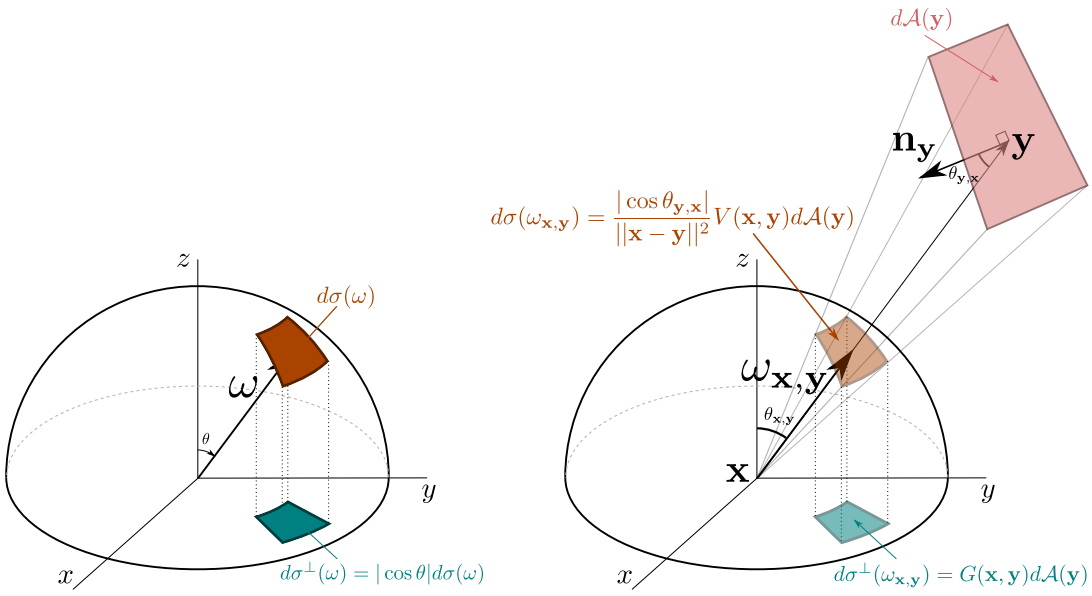


Figure 6 – Differential projected solid angle measure and differential area measure.

This integral can be expressed directly on surfaces of the scene using the change of variables $\mathbf{y} = \mathbf{x}_\mathcal{M}(\omega_{\mathbf{x},\mathbf{y}})$ and:

$$d\sigma(\omega_{\mathbf{x},\mathbf{y}}) = \frac{|\cos\theta_{\mathbf{y},\mathbf{x}}|}{\|\mathbf{x}-\mathbf{y}\|^2} V(\mathbf{x},\mathbf{y}) d\mathcal{A}(\mathbf{y}) \tag{16}$$

The integral can then be expressed with respect to area measure (Figure 6):

$$\begin{aligned} F &= \int_{\mathbf{y}\in\mathcal{M}} f(\mathbf{y},\omega_{\mathbf{y},\mathbf{x}}) \frac{|\cos\theta_{\mathbf{x},\mathbf{y}}||\cos\theta_{\mathbf{y},\mathbf{x}}|}{\|\mathbf{x}-\mathbf{y}\|^2} V(\mathbf{x},\mathbf{y}) d\mathcal{A}(\mathbf{y}) \\ &= \int_{\mathbf{y}\in\mathcal{M}} f(\mathbf{y},\omega_{\mathbf{y},\mathbf{x}}) G(\mathbf{x},\mathbf{y}) d\mathcal{A}(\mathbf{y}) \end{aligned} \tag{17}$$

In this equation, $G(\mathbf{x},\mathbf{y})$ is the *geometric factor* between the points $\mathbf{x}$ and $\mathbf{y}$ and we have:

$$d\sigma^\perp_\mathbf{x}(\omega_{\mathbf{x},\mathbf{y}}) = G(\mathbf{x},\mathbf{y}) d\mathcal{A}(\mathbf{y}) \tag{18}$$

2.2 RADIOMETRIC QUANTITIES

Radiometry is the branch of physics that defines quantities related to electromagnetic radiation by integration of a density function called *radiance*, which represents the distribution of radiation's energy in a scene. Radiance has to be estimated by rendering algorithms to produce digital images of a virtual scene.

2.2.1 *Radiance*

Radiance describes the density of visible light energy leaving a surface point $\mathbf{x} \in \mathcal{M}$ in a direction $\omega_o \in \mathcal{S}^2$ at a particular instant of time $t \in \mathbb{R}$ and for a visible wavelength $w \in [380, 750]$.

Radiance can then be described by the *spectral radiance function* $L_\lambda : \mathcal{M} \times \mathcal{S}^2 \times \mathbb{R} \times [380, 750] \to \mathbb{R}$, a density function that can be integrated in order to express other radiometric quantities such as radiant power. The human visual system is preferentially sensitive to red, green and blue, thus any visible color can be represented by a vector of three scalar values. Consequently, the wavelength dependency of radiance is generally ignored and the spectral radiance function is replaced by three *radiance functions* L_R, L_G and L_B. Each of these functions is obtained by integration of the product between spectral radiance and the CIE RGB spectral sensitivity functions R, G and B, such that:

$$\forall C \in \{R, G, B\}, \qquad L_C(\mathbf{x}, \omega_o, t) := \int_{380}^{750} C(w) L_\lambda(\mathbf{x}, \omega_o, t, w) dw \tag{19}$$

With this definition, rendering a color image is equivalent to rendering three monochromatic images corresponding to the red, green and blue channel of the image.

Radiometry assumes that the speed of light is infinite. Consequently, we consider that the radiance field reaches its equilibrium instantly in a static scene. An animated scene can be seen as a sequence of static scenes, each one having its own radiance field at a time $t_i \in \mathbb{R}$, $i \in \mathbb{N}$. Thus, for conciseness, the time dependency of radiance is often made implicit and the parameter t is dropped from the notation. Note, however, that to formally define time dependent effects, such as *motion blur*, time must be taken into account and reintroduced in the notation.

We now consider a single radiance function $L : \mathcal{R}_\mathcal{M} \to \mathbb{R}$. The quantity $L(\mathbf{x}, \omega_o) \in \mathbb{R}$ is the radiance leaving point $\mathbf{x} \in \mathcal{M}$ in direction $\omega_o \in \mathcal{S}^2$ and is expressed in watt per steradian per square metre per second [$\mathrm{W \cdot m^{-2} \cdot sr^{-1} \cdot s^{-1}}$]. For simplicity, we also use the notation $L(\mathbf{x}, \mathbf{y})$ referring to $L(\mathbf{x}, \omega_{\mathbf{x},\mathbf{y}})$ with $\mathbf{x}, \mathbf{y} \in \mathcal{M}$.

Radiance is assumed to be constant along straight lines in a vacuum, such that we can define the radiance $L_i(\mathbf{x}, \omega_i)$ received by a point $\mathbf{x}$ from an incident direction ω_i as:

$$L_i(\mathbf{x}, \omega_i) := \begin{cases} L(\mathbf{x}_\mathcal{M}(\omega_i), -\omega_i) \text{ if } d_\mathcal{M}(\mathbf{x}, \omega_i) < +\infty \\ 0 \text{ otherwise.} \end{cases} \tag{20}$$

L_i is referred as the *incident radiance function* (Figure 7).

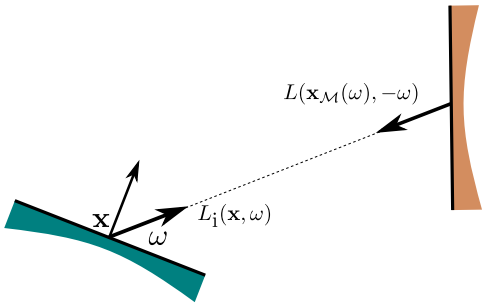


Figure 7 – Incident radiance function.

2.2.2 *Radiant power*

By integration of radiance, we can express the amount of light energy leaving a surface $\mathcal{M}_i$ per unit of time. This quantity $\Phi(\mathcal{M}_i)$ is referred as *exitant radiant power* and defined by:

$$\Phi(\mathcal{M}_i) \coloneqq \int_{\mathbf{x}\in\mathcal{M}_i}\int_{\omega_\mathrm{o}\in\mathcal{S}^2} L(\mathbf{x},\omega_\mathrm{o})d\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{o})d\mathcal{A}(\mathbf{x}) \tag{21}$$

Similarly, the *incident radiant power* $\Phi_\mathrm{i}(\mathcal{M}_i)$, reaching the surface is defined by:

$$\Phi_\mathrm{i}(\mathcal{M}_i) \coloneqq \int_{\mathbf{x}\in\mathcal{M}_i}\int_{\omega_\mathrm{i}\in\mathcal{S}^2} L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})d\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{i})d\mathcal{A}(\mathbf{x}) \tag{22}$$

These definitions lead to an expression of radiance as the derivative of exitant radiant power:

$$L(\mathbf{x},\omega_\mathrm{o}) = \frac{\partial^2\Phi(\mathbf{x},\omega_\mathrm{o})}{\partial\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{o})\partial\mathcal{A}(\mathbf{x})} \tag{23}$$

and similarly for incident radiance:

$$L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i}) = \frac{\partial^2\Phi_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})}{\partial\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{o})\partial\mathcal{A}(\mathbf{x})} \tag{24}$$

2.2.3 *Irradiance and radiosity*

Irradiance E describes the amount of radiant power per square meter received by a surface point:

$$E(\mathbf{x}) \coloneqq \int_{\omega_\mathrm{i}\in\mathcal{S}^2} L_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})d\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{i}) \tag{25}$$

$$= \frac{\partial\Phi_\mathrm{i}(\mathbf{x},\omega_\mathrm{i})}{\partial\mathcal{A}(\mathbf{x})} \tag{26}$$

The corresponding quantity for radiant power per square meter leaving the surface point is *radiosity B*:

$$B(\mathbf{x}) \coloneqq \int_{\omega_\mathrm{o}\in\mathcal{S}^2} L(\mathbf{x},\omega_\mathrm{o})d\sigma^\perp_{\mathbf{x}}(\omega_\mathrm{o}) \tag{27}$$

$$= \frac{\partial\Phi(\mathbf{x},\omega_\mathrm{o})}{\partial\mathcal{A}(\mathbf{x})} \tag{28}$$

These two quantities are mostly manipulated by light transport simulation involving lambertian materials (see Section 2.3), for which the amount of reflected light is independent on the incident or exitant direction.

2.2.4 *Emission*

Emission of radiant power in the scene is represented by the *emitted radiance function* $L_\mathrm{e} : \mathcal{R}_\mathcal{M} \to \mathbb{R}$. In practice, this function is given with the input scene. Each surface $\mathcal{M}_i$ is

associated to a partial emitted radiance function $L_e^{(i)} : \mathcal{R}_{\mathcal{M}} \to \mathbb{R}$ which is null at rays having their origin outside $\mathcal{M}_i$. We define the emitted radiance function as:

$$L_e(\mathbf{x}, \omega_o) := \sum_{i=1}^{N_{\mathcal{M}}} L_e^{(i)}(\mathbf{x}, \omega_o) \tag{29}$$

A *light source* is a surface $\mathcal{M}_i$ such that the partial emitted radiance function $L_e^{(i)}$ is not null everywhere. We denote by $\mathcal{M}_L \subset \mathcal{M}$ the union of all light sources.

To simplify the mathematical formulation presented in this thesis, this definition of emitted radiance does not take into account non physical light sources such as point sources, directional sources and environment sources, that are often included in rendering engines. These kind of light sources can be defined by expending the domain of the emitted radiance function to $\mathcal{R}$ (for point sources) and $\mathcal{S}^2$ (for directional and environment sources) . The emitted radiance function of point and directional light sources must include a Dirac distribution to account for the geometric singularity. The change of integration measure between the light source domain and projected solid angle must account for the geometric structure of the light source.

2.3 SURFACE SCATTERING

Scattering describes how light interacts with matter, i.e. how photons deviate from their trajectory and how their energy is transformed. In this section, we present a surface scattering model mostly used in computer graphics to describe and simulate material appearance. This model is based on the *bidirectional scattering distribution function* (BSDF) [Vea97, p. 85] that describes how each material scatters incident light. Material appearance is a very active area of research, aiming at representing more realistic materials through the development of advanced BSDF models.

2.3.1 *The bidirectional scattering distribution function*

For light transport simulation without participating medium, it is sufficient to describe how light is reflected or refracted at surfaces of the scene. This interaction is modeled by the *bidirectional scattering distribution function* (BSDF) $f_s : \mathcal{M} \times \mathcal{S}^2 \times \mathcal{S}^2 \to \mathbb{R}$. For a surface point $\mathbf{x} \in \mathcal{M}$, an incident direction $\omega_i \in \mathcal{S}^2$ and an outgoing direction $\omega_o \in \mathcal{S}^2$, $f_s(\mathbf{x}, \omega_i, \omega_o)$ is the proportion of radiance incident from ω_i that is scattered toward ω_o at the point $\mathbf{x}$:

$$f_s(\mathbf{x}, \omega_i, \omega_o) := \frac{dL_s(\mathbf{x}, \omega_o)}{L_i(\mathbf{x}, \omega_i) d\sigma_{\mathbf{x}}^{\perp}(\omega_i)} \tag{30}$$

where $L_s = L - L_e$ is the *scattered radiance function*.

In practice, the BSDF is given with the scene as an input of the rendering algorithm. Each surface has an associated material which defines the BSDF at each point of the surface.

For physically based scattering, the BSDF must respect the following properties:

Helmotz reciprocity states that the behavior of an homogeneous material is the same when directions are exchanged:

$$f_s(\mathbf{x}, \omega_i, \omega_o) = f_s(\mathbf{x}, \omega_o, \omega_i) \tag{31}$$

Energy conservation ensures that a material does not scatter more energy than it receives:

$$\forall \omega_i \in \mathcal{S}^2, \int_{\omega_o \in \mathcal{S}^2} f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma_{\mathbf{x}}^{\perp}(\omega_o) \leq 1 \tag{32}$$

Given three points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{M}$, we also employ the notation:

$$f_s(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) := f_s(\mathbf{x}_2, \omega_{\mathbf{x}_2, \mathbf{x}_1}, \omega_{\mathbf{x}_2, \mathbf{x}_3}) \tag{33}$$

2.3.2 *Scattered radiance*

From Equation (30), we obtain an integral expression for the radiance $L_s(\mathbf{x}, \omega_o)$ scattered by a surface point $\mathbf{x}$ in a direction ω_o:

$$L_s(\mathbf{x}, \omega_o) = \int_{\omega_i \in \mathcal{S}^2} f_s(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) d\sigma_{\mathbf{x}}^{\perp}(\omega_i) \tag{34}$$

This expression is involved in the *rendering equation* (37) defined in Section 2.4.3.

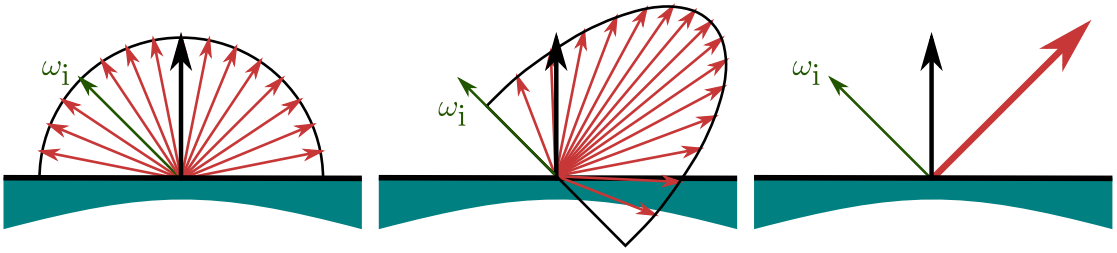
2.3.3 *BSDF models*

Three types of scattering phenomena are generally simulated by BSDFs (Figure 8):

— *diffuse scattering* does not depend on the incident direction.

— *glossy scattering* scatters light in preferred directions that depend on the incident direction.

— *specular scattering* scatters light in only two directions: the perfect reflexion direction and the perfect transmission direction.



(a) Diffuse reflection (b) Glossy reflection (c) Specular reflection

Figure 8 – Illustration of the BSDF for diffuse, glossy and specular reflections. The length of the arrows represent the magnitude of the BSDF for different outgoing directions. For specular reflection, this magnitude cannot be represented by a real number, but involves a Dirac distribution.

Many BSDF models have been developed to approximate surface scattering with various degrees of realism. The most simple ones are *empirical models*, such as the Lambert BSDF, for ideal diffuse materials, or the Phong BSDF [LW94], for glossy materials. These BSDFs are expressed by simple expressions that are fast to evaluate and easy to sample analytically. However, they are not physically based and introduce a non-negligible error in the appearance of materials compared to real materials. *Measured models* [MPBM03] are based on a complete tabulation of the BSDF for each material, obtained from measurements of real materials. These kind of BSDF are accurate since they are based on real data, but involve high memory requirement due to the storage of measurements. *Physically based models*, such as microfacet BSDFs [Hei14], are derived mathematically and provide better accuracy for the simulation of complex materials. They are generally more difficult to sample than empirical models. Specular scattering is simply derived from *Snell's law* of refraction and reflection and lead to a Dirac distribution in the BSDF expression for proper integration in the BSDF framework.

2.4 LIGHT TRANSPORT EQUATIONS

Rendering an image requires to compute measurements on radiance incident to the virtual camera. These measurements are expressed by the *measurement equation* (Section 2.4.1) that relates the intensity of each pixel to the incident radiance field. The *rendering equation* (Section 2.4.3) gives an expression for the radiance at each ray. Unfortunately, these two equations can not be solved analytically and numerical estimation methods must be used (see Chapter 3).

2.4.1 *The measurement equation*

To render an image, each pixel, identified by an index i, is associated to a *sensor response function* $W_e^{(i)} : \mathcal{R}_\mathcal{M} \to \mathbb{R}$ such that the following equation expresses the intensity of the pixel:

$$\begin{aligned} I_i &= \int_{(\mathbf{x},\omega_i)\in\mathcal{R}_\mathcal{M}} W_e^{(i)}(\mathbf{x},\omega_i) L_i(\mathbf{x},\omega_i) d\sigma_\mathbf{x}^\perp(\omega_i) d\mathcal{A}\mathbf{x} \\ &= \int_{(\mathbf{x}_1,\mathbf{x}_2)\in\mathcal{M}^2} W_e^{(i)}(\mathbf{x}_1,\mathbf{x}_2) G(\mathbf{x}_1,\mathbf{x}_2) L(\mathbf{x}_2,\mathbf{x}_1) d\mathcal{A}(\mathbf{x}_1) d\mathcal{A}(\mathbf{x}_2) \end{aligned} \tag{35}$$

This equation is referred as *the measurement equation*.

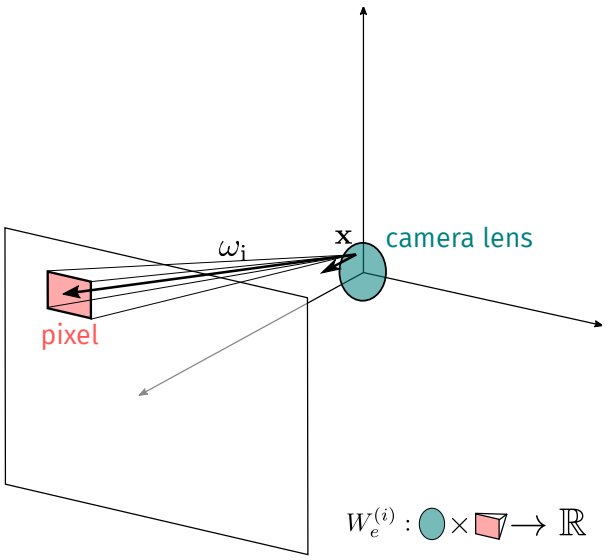


Figure 9 – Domain of the sensor response function for a perspective camera and a pixel.

2.4.2 *Sensor response*

Most often, the virtual camera is represented by a lens $\mathcal{L} \subset \mathcal{M}$ and the function $W_e^{(i)}$ is not null only for rays starting at $\mathcal{L}$ and pointing to the image plane (Figure 9). More specifically, $W_e^{(i)}(\mathbf{x}, \omega_i)$ takes the form:

$$W_e^{(i)}(\mathbf{x}, \omega_i) = h_i(u, v) W_e(\mathbf{x}, \omega_i) \tag{36}$$

where (u, v) are the image plane coordinates of the ray $(\mathbf{x}, \omega_i)$, h_i is an image filter function (for example a box filter or a gaussian filter), and $W_e(\mathbf{x}, \omega_i)$ is a sensor response function that only depends on the camera geometry. This decomposition is useful when the pixel filter function is such that a ray can contribute to several pixel intensities, in order to share the computation between pixels.

The measurement equation (35) is quite general and can be used to express measurements other than pixel intensities. For example, the irradiance at a surface point $\mathbf{x}_0 \in \mathcal{M}$ can be expressed by the measurement equation with the sensor response function $W_e(\mathbf{x}, \omega_i) = \delta_{\mathcal{A}}(\mathbf{x} - \mathbf{x}_0)$, where $\delta_{\mathcal{A}}$ is the Dirac distribution with respect to area measure.

The sensor response function is also called the *emitted importance function* and $W_e^{(i)}(\mathbf{x}, \omega)$ is the importance emitted at the ray $(\mathbf{x}, \omega)$ by sensor i. We note $\mathcal{M}_W^{(i)} \subset \mathcal{M}$ the union of all surface points corresponding to the sensor i, that is $\forall \mathbf{x} \in \mathcal{M}_W^{(i)}$, $\exists \omega \in \mathcal{S}^2$ such that $W_e^{(i)}(\mathbf{x}, \omega) > 0$. The union of all surfaces corresponding to sensors is denoted by $\mathcal{M}_W := \bigcup_{i=1}^{N_{\mathcal{M}_W}} \mathcal{M}_W^{(i)}$, where $N_{\mathcal{M}_W}$ is the number of sensors defined for the simulation.

2.4.3 *The rendering equation*

The main difficulty for the evaluation of the measurement equation (35) is the radiance factor in the integrand. The radiance function is the solution of the *rendering equation* [Kaj86], defined by:

$$\begin{aligned} L(\mathbf{x}, \omega_o) &= L_e(\mathbf{x}, \omega_o) + L_s(\mathbf{x}, \omega_o) \\ &= L_e(\mathbf{x}, \omega_o) + \int_{\omega_i \in \mathcal{S}^2} L_i(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma^{\perp}_{\mathbf{x}}(\omega_i) \end{aligned} \tag{37}$$

It is not possible to solve this equation analytically for general scenes. Therefore, radiance can only be estimated using numerical integration.

2.4.4 *Path integral formulation of light transport*

The path integral framework, introduced by Veach [Vea97], expresses the measurement equation by an integral over the *path space* $\mathcal{P}$:

$$I_i = \int_{\bar{\mathbf{x}} \in \mathcal{P}} f^{(i)}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \tag{38}$$

where $f^{(i)}$ is the *path contribution function* for the sensor i. This formulation is well suited to express rendering algorithms based on path sampling (Chapter 4).

A path $\bar{\mathbf{x}} = \mathbf{x}_1 ... \mathbf{x}_k \in \mathcal{P}$ (Figure 10) is a sequence of surface points $\mathbf{x}_i \in \mathcal{M}$ referred as *vertices*. The *length* $k \geq 2$ is the number of vertices of the path. The space of paths of length k is denoted by $\mathcal{P}_k$ and we have:

$$\mathcal{P} := \bigcup_{k \geq 2} \mathcal{P}_k \tag{39}$$

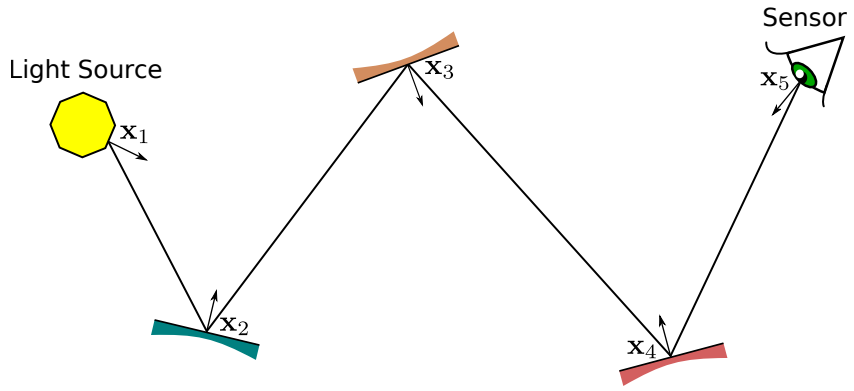


Figure 10 – Illustration of a path $\bar{\mathbf{x}} = \mathbf{x}_1 .. \mathbf{x}_5$ from $\mathcal{P}_5$.

The differential measure $d\mu(\bar{\mathbf{x}})$ of a path is defined as the product of differential area measures:

$$d\mu(\mathbf{x}_1 ... \mathbf{x}_k) := \prod_{i=1}^{k} dA(\mathbf{x}_i) \tag{40}$$

The contribution of an individual path can be derived by expending recursively the definition of $L(\mathbf{x}_2, \mathbf{x}_1)$ in the measurement equation (35) using the rendering equation (37), to obtain:

$$\begin{aligned} f^{(i)}(\mathbf{x}_1 ... \mathbf{x}_k) &= L_\mathrm{e}(\mathbf{x}_1, \mathbf{x}_2) T(\mathbf{x}_1 ... \mathbf{x}_k) W_\mathrm{e}^{(i)}(\mathbf{x}_k, \mathbf{x}_{k-1}) \\ T(\mathbf{x}_1 ... \mathbf{x}_k) &= \left[\prod_{i=1}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) \prod_{i=2}^{k-1} f_\mathrm{s}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \right] \end{aligned} \tag{41}$$

The contributions of paths of length k is noted $I_{i,k}$, such that:

$$I_{i,k} = \int_{\bar{\mathbf{x}} \in \mathcal{P}_k} f_i(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \tag{42}$$

$$I_i = \sum_{k \geq 2} I_{i,k} \tag{43}$$

3

MONTE CARLO INTEGRATION

As described in Chapter 2, a rendering algorithm has to solve the measurement equation for each pixel of an image to render. This equation is expressed by an integral that involves the radiance function, which is itself the solution of the rendering equation. The rendering equation is an integral equation that cannot be solved analytically for general scenes. Consequently, rendering algorithms have to estimate the solution of the measurement equation for each pixel using numerical methods. To be viable for light transport simulation, an estimator has to be *consistent*, meaning that it must converge to the estimated value over time. Monte Carlo (MC) integration [MU49] satisfies that criterion and is the most widely used estimation method for global illumination. The reason for this is that the convergence rate of MC integration does not depend on the dimension of the integration domain, which is infinite in the case of the path space. MC integration is a stochastic estimation strategy that relies on probability theory and more particularly on the law of large numbers. This chapter gives an overview of MC integration as well as common strategies to improve the convergence speed of the estimation.

Section 3.1 reviews material related to probability theory and random sampling required to define a practical MC estimator. The basic shape of MC estimators is presented in Section 3.2. In Section 3.3, we introduce several common methods to reduce the variance of MC estimation, since the error produced by a MC estimator is directly related to its variance.

Contents

3.1 PROBABILITY REVIEW

3.1.1 *Random variables*

A *random variable* X, taking values in a *sample space* Ω, represents a process producing random outcomes in Ω with some probability measure P_X defined on *events*. An event is a subset of Ω and the set of events, noted $\mathcal{E}_X$, must define a *sigma-algebra* on Ω, that is:

— $\Omega \in \mathcal{E}_X$: the whole sample space is an event
— $\forall A \in \mathcal{E}_X$, $\bar{A} = \Omega \setminus A \in \mathcal{E}_X$: every event has a complementary event
— For any countable sequence $(A_i)_{i\in\mathbb{N}}$ of events, $\cup_{i\in\mathbb{N}} A_i \in \mathcal{E}_X$: a countable union of events is also an event

An event A is said to be observed if the random realization of X belongs to A. The *impossible event* is $\emptyset = \Omega^c$, the *certain event* is Ω. For two events $A, B \in \mathcal{E}_X$, $A \cup B$ represents the event "A is observed or B is observed" and similarly $A \cap B$ represents the event "A is observed and B is observed". Two events A, B are said to be *incompatible* if $A \cap B = \emptyset$.

The probability measure $P_X : \mathcal{E} \to [0, 1]$ associates a probability $P_X(E)$ of being observed to each event E, also noted $P(X \in E)$. The probability measure must satisfy the following conditions:

— $P_X(\Omega) = 1$: there is a 100% chance of observing an outcome in Ω.
— For any sequence $(A_i)_{i\in\mathbb{N}}$ of mutually incompatible events we have $P_X(\cup_{i\in\mathbb{N}} A_i) = \sum_{i\in\mathbb{N}} P_X(A_i)$

Note that, since $A \cap \bar{A} = \emptyset$, we have $P_X(\bar{A}) = 1 - P_X(A)$.

For numerical simulations, random numbers are generated on $[0, 1]$ by a pseudo-random number generator (PRNG) with uniform probability. Even if a computer is not able to generate every possible real number in $[0, 1]$, we consider that the outcomes of a PNRG are distributed according to a random variable $\mathcal{U}$, with the set of events being represented as the smallest sigma-algebra containing all the intervals of $[0, 1]$, referred as the *borelian sigma-algebra*. The probability measure $P_{\mathcal{U}}$ is uniform, that is:

$$\forall a, b \in [0, 1], a \leq b \implies P_{\mathcal{U}}([a, b]) = b - a \tag{44}$$

We note $\varepsilon \sim \mathcal{U}([0, 1])$ if ε is a realization distributed uniformly on $[0, 1]$.

3.1.2 *Continuous random variables*

Continuous random variables take their values in uncountable sample spaces, such as the real numbers $\mathbb{R}$ or the n-dimensional vectors $\mathbb{R}^n$. In rendering, we are interested in geometric sample spaces, like the sphere of directions $\mathcal{S}^2$, the surfaces of a scene $\mathcal{M}$ or the path space $\mathcal{P}$, on which we can define random variables by transforming random variables of $\mathbb{R}^n$.

3.1.2.1 *Real-valued random variables*

A real-valued random variable X takes values in the real numbers $\mathbb{R}$. The cumulative distribution function (cdf) F_X of a real-valued random variable X is defined by:

$$\begin{aligned} F_X(a) &:= P_X((-\infty, a]) \\ &:= P(X \leq a) \end{aligned} \tag{45}$$

Under certain conditions, the distribution of X can be described by the probability density function (pdf) p_X such that:

$$F_X(a) = \int_{-\infty}^{a} p_X(x)dx \Leftrightarrow p_X = \frac{dF_X}{dx} \tag{46}$$

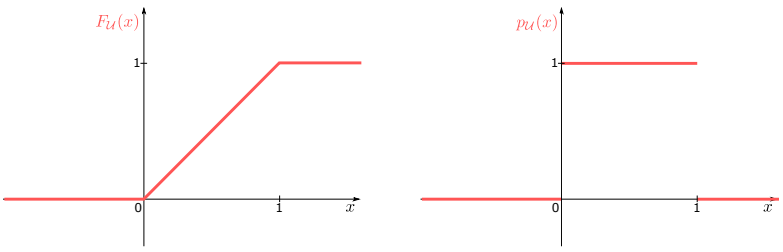
For the remaining of this thesis, we only consider continuous random variables that can be described by a pdf.

Example 3.1. *The cdf of a uniform random variable $\mathcal{U}$ on $[0,1]$ is given by (Figure 11):*

$$F_{\mathcal{U}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \in [0,1] \\ 1 & \text{if } x > 1 \end{cases} \tag{47}$$

and its pdf:

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & \text{if } x \in [0,1] \\ 0 & \text{otherwise.} \end{cases} \tag{48}$$



(a) uniform cdf (b) uniform pdf

Figure 11 – Illustration of the curves associated to the uniform distribution $\mathcal{U}([0,1])$.

3.1.2.2 *Real-valued random vectors*

A random vector $X = (X_1, ..., X_n)$ takes values in $\mathbb{R}^n$. The *joint cdf* F_X of X is defined by:

$$\begin{aligned} F_X(a_1, ..., a_n) &:= P_X((-\infty, a_1] \times ... \times (-\infty, a_n]) \\ &:= P(X_1 \leq a_1, ..., X_n \leq a_n) \end{aligned} \tag{49}$$

and the corresponding *joint pdf* p_X is such that:

$$F_X(a_1, ..., a_n) = \int_{-\infty}^{a_1} ... \int_{-\infty}^{a_n} p_X(x_1, ..., x_n)dx_1...dx_n \Leftrightarrow p_X = \frac{d^n F_X}{dx_1...dx_n} \tag{50}$$

The *marginal pdf* p_{X_i} of a single variable X_i from the vector X is obtained by integrating out the remaining variables:

$$p_{X_i}(x_i) := \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} p_X(x_1, ..., x_n) dx_1 ... dx_{i-1} dx_{i+1} ... dx_n \tag{51}$$

More generally, the marginal pdf $p_{X_{i_1},...,X_{i_k}}$ of any sub-vector $(X_{i_1}, ..., X_{i_k})$ of $(X_1, ..., X_n)$ is obtained by integrating out the $n - k$ random variables not included in the sub-vector.

The *conditional pdf* $p_{X_1,...,X_k|X_{k+1},...,X_n}$ of the sub-vector $(X_1, ..., X_k)$ given that $X_{k+1} = x_{k+1}, ..., X_n = x_n$ is defined by:

$$p_{X_1,...,X_k|X_{k+1},...,X_n}(x_1, ..., x_k|x_{k+1}, ..., x_n) := \frac{p_X(x_1, ..., x_n)}{p_{X_{k+1},...,X_n}(x_{k+1}, ..., x_n)} \tag{52}$$

Note that this definition can be generalized for any sub-vector $(X_{i_1}, ..., X_{i_k})$, given the value of the remaining $n - k$ random variables.

The definition of the conditional pdf leads to the following relationship between the joint, marginal and conditional pdfs:

$$p_X(x_1, ..., x_n) = p_{X_1}(x_1) p_{X_2|X_1}(x_2|x_1) ... p_{X_n|X_1,...,X_{n-1}}(x_n|x_1, ..., x_{n-1}) \tag{53}$$

This relation can be used to sample random vectors according to some specified joint pdf, as described in Section 3.1.4.2.

When the random variables $X_1, ..., X_n$ are independent, Equation (53) simplifies to:

$$p_X(x_1, ..., x_n) = \prod_{i=1}^{n} p_{X_i}(x_i) \tag{54}$$

The probability of any event $A \subset \mathbb{R}^n$ can be expressed by integrating the pdf over A:

$$P(X \in A) = \int_{x \in A} p_X(x) dx \tag{55}$$

We note $X \sim F$ or $X \sim p$ when a random variable X is distributed according to a given cdf F or pdf p.

3.1.3 *Transformation of continuous random variables*

We saw on Chapter 2 that light transport integrals are expressed over geometric domains. To distribute random elements on such domains, we sample $[0, 1]^n$ and we apply a bijective transformation $Y : [0, 1]^n \to \Omega$ where Ω is the integration domain of interest. If X is a random vector from $[0, 1]^n$, then $Y(X)$ is a new random variable with pdf p_Y satisfying:

$$\forall A \subset \Omega, P_Y(A) = \int_{y \in A} p_Y(y) d\mu(y) \tag{56}$$

where μ is a measure on Ω.

To obtain an expression for $p_Y(y)$, we apply the change of variable $x = Y^{-1}(y)$:

$$P_Y(A) = \int_{x \in Y^{-1}(A)} p_Y(Y(x)) |J_Y(x)| dx \tag{57}$$

where $J_Y(x)$ is the jacobian of transformation Y evaluated at x.

By definition of $P_Y(A)$, we also have:

$$P_Y(A) = P(Y \in A) = P(X \in Y^{-1}(A)) = P_X(Y^{-1}(A)) = \int_{x \in Y^{-1}(A)} p_X(x) dx \tag{58}$$

Since the equality must hold for any subset of Ω, we have:

$$p_Y(Y(x)).|J_Y(x)| = p_X(x) \Leftrightarrow p_Y(Y(x)) = \frac{p_X(x)}{|J_Y(x)|} \tag{59}$$

Example 3.2. *We want to sample the unit sphere $\mathcal{S}^2$. For that, we can sample $(u, v) \sim \mathcal{U}([0,1]^2)$ and apply the transformation $\omega = Y(u, v) = (\sin(\pi u)\cos(2\pi v), \sin(\pi u)\sin(2\pi v), \cos(\pi u))$. The jacobian of this transformation is $J_Y(u, v) = 2\pi^2 \sin(\pi u)$. Consequently, the pdf of Y is:*

$$p_Y(\omega) = \frac{p_{\mathcal{U}}(u, v)}{|2\pi^2 \sin(\pi u)|} = \frac{1}{2\pi^2 |\sin(\pi u)|} \tag{60}$$

We observe that this pdf is not the uniform pdf $p(\omega) = \frac{1}{4\pi}$ on $\mathcal{S}^2$. The next section details how to sample random variables distributed according to a specified pdf.

3.1.4 *Sampling continuous random variables*

Most often, we need to sample random elements from Ω distributed according to an importance function $f : \Omega \to \mathbb{R}$. The associated pdf p is the normalization of f defined by $p(x) = f(x) / \int_\Omega f(x) d\mu(x)$, such that p is proportional to f and integrate to one over the whole domain. A comprehensive reference for the generation of non-uniform random variables is provided by the book "Non-Uniform Random Variate Generation" [Dev86]. In this section, we describe *inverse transform sampling*, a method allowing to sample the domain when the pdf can be integrated to obtain the cdf, and when this cdf can be analytically inverted. When this requirements are satisfied, we can sample uniformly $[0,1]^n$ and apply the inverted cdf to the result in order to obtain elements distributed according to the pdf.

3.1.4.1 *Dimension 1.*

We want to sample random real values according to a cdf $F_X : \mathbb{R} \to [0,1]$. For a uniform random variable $U \sim \mathcal{U}([0,1])$, we set $X := F_X^{-1}(U)$ and we have $X \sim F_X$. Indeed:

$$P(X \leq a) = P(F_X^{-1}(U) \leq a) = P(U \leq F_X(a)) = F_{\mathcal{U}}(F_X(a)) = F_X(a) \tag{61}$$

3.1.4.2 *Dimension n.*

We want to sample random real vectors according to a joint cdf $F_{X_1,...,X_n}$. For that, we sample iteratively each component of the vector according to the marginal/conditional pdfs which are all 1-dimensional. More specifically, given n independent uniform random variables $U_1, ..., U_n \sim \mathcal{U}([0,1]^n)$, we set:

$$\begin{aligned} X_1 &= F^{-1}_{X_1}(U_1) \\ X_2 &= F^{-1}_{X_2|X_1}(U_2|X_1) \\ &... \\ X_n &= F^{-1}_{X_n|X_1,...,X_{n-1}}(U_n|X_1, ..., X_{n-1}) \end{aligned} \tag{62}$$

From Equation (53) we have $(X_1, ..., X_n) \sim F_{X_1,...,X_n}$

3.1.4.3 *General domains.*

For a general continuous domain Ω, we sample a random vector $X \in [0,1]^n$ according to the pdf $p_X(x) = p(\varphi(x))|J_\varphi(x)|$ where φ is an arbitrary bijective transformation from $[0,1]^n$ to Ω. From Equation (59), we have $\varphi(X) \sim p$.

3.1.5 *Discrete random variables*

The notion of discrete random variables is useful to select an element from a finite set. For example, choosing a light source according to the emitted power of each one.

The probability distribution of a discrete random variable $X \in \Omega$, where Ω is a countable set of elements (finite or not), is characterized by its probability mass function (pmf) $p_X : \Omega \to [0,1]$:

$$\begin{aligned} p_X(a_n) &= P(X = a_n) \\ \text{with the condition } &\sum_{a_n \in \Omega} p_X(a_n) = 1 \end{aligned} \tag{63}$$

The probability mass function is the discrete analog of the probability density function for continuous random variables. It is possible to define a pmf of a discrete random variable as a pdf on $\mathbb{R}$ using Dirac distributions:

$$p_X(x) = \sum_{a_n \in \Omega} P(X = a_n)\delta(x - a_n) \tag{64}$$

The cumulative distribution function F_X of a discrete random variable X is defined similarly than the continuous case, but the integral is transformed into a sum due to Dirac distributions:

$$F_X(a) = P(X \leq a) = \int_{-\infty}^{a} \Big(\sum_{a_n \in \Omega} P(X = a_n)\delta(x - a_n) \Big) dx = \sum_{a_n \leq a} P(X = a_n) \tag{65}$$

3.1.6 *Sampling discrete random variables*

Sampling random variables according to a discrete distribution can be done by tabulating the cdf, drawing a uniformly distributed random number $\varepsilon \in [0, 1)$ and searching in the tabulated cdf the first index i such that $\varepsilon \leq F_X(i)$. Since the cdf is monotonically increasing, the tabulated cdf is sorted and the search can be done in $\mathcal{O}(\log_2 n)$ time using binary search. This procedure is the discrete equivalent of inverse transform sampling.

3.1.7 *Expected value and variance*

The *expected value* (also denoted by *mean* or *expectation*) of a random variable $X \in \Omega$ is defined by:

$$\mathbb{E}[X] := \int_{x \in \Omega} x p_X(x) dx \tag{66}$$

If $f : \Omega \to \Omega'$ is a function from Ω to Ω', then $Y = f(X)$ is random variable with expected value:

$$\mathbb{E}[Y] := \int_{x \in \Omega} f(x) p_X(x) dx \tag{67}$$

A useful property of the expected value is linearity.

$$\begin{aligned} \mathbb{E}[X_1 + X_2] &:= \mathbb{E}[X_1] + \mathbb{E}[X_2] \\ \forall a \in \mathbb{R},\ \mathbb{E}[aX] &= a\mathbb{E}[X] \end{aligned} \tag{68}$$

When X_1 and X_2 are independent, we also have:

$$\mathbb{E}[X_1 X_2] := \mathbb{E}[X_1]\mathbb{E}[X_2] \tag{69}$$

The variance of Y is defined by:

$$\mathbb{V}[Y] := \mathbb{E}[(\mathbb{E}[Y] - Y)^2] = \int_{x \in \Omega} (\mathbb{E}[Y] - f(x))^2 p_X(x) dx \tag{70}$$

Another expression for the variance is given by:

$$\mathbb{V}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \tag{71}$$

which can be obtained by applying the linearity property of the expected value.

If X_1 and X_2 are independent, we have the linearity of the variance:

$$\mathbb{V}[X_1 + X_2] := \mathbb{V}[X_1] + \mathbb{V}[X_2] \tag{72}$$

For $a \in \mathbb{R}$, we have:

$$\mathbb{V}[aY] = a^2 \mathbb{V}[Y] \tag{73}$$

3.1.8 *Laws of large numbers*

Let $(X_i)_{i\in\mathbb{N}}$ be an infinite sequence of independent and identically distributed (i.i.d) random variables with expected value $\mu = \mathbb{E}[X_i]$. The sample average $\bar{X}_n$ of the sequence is defined by:

$$\bar{X}_n = \frac{X_1 + ... + X_n}{n} \tag{74}$$

The *weak law of large numbers* states that for high values of n, the sample average is close to μ with high probability:

$$\forall \epsilon > 0, \lim_{n\to\infty} P(|\bar{X}_n - \mu| \leq \epsilon) = 1 \tag{75}$$

We say that $\bar{X}_n$ *converges in probability* to μ.

The *strong law of large numbers* is a stronger convergence property, in the sense that it implies the weak law, given by:

$$P(\lim_{n\to\infty} \bar{X}_n = \mu) = 1 \tag{76}$$

We say that $\bar{X}_n$ *converges almost surely* to μ.

These two laws state that we can use the sample average as a consistent and unbiased estimator of μ, which is the purpose of MC estimation.

3.2 MONTE CARLO ESTIMATION

3.2.1 *Estimation of an integral*

Given a function $f : \Omega \to \mathbb{R}$, we want to estimate the following integral:

$$I := \int_{x\in\Omega} f(x)dx \tag{77}$$

Let X be a random variable on Ω distributed according to a probability density function $p : \Omega \to \mathbb{R}$ such that $f(x) \neq 0 \Rightarrow p(x) \neq 0$ (every element contributing to the integral can be sampled according to p). The integral can then be rewritten as:

$$I = \int_{x\in\Omega} \frac{f(x)}{p(x)} p(x)dx \tag{78}$$

showing that $I = \mathbb{E}[\frac{f(X)}{p(X)}]$. Given $x \in \Omega$ a sample distributed according to p, the following expression is a *primary estimator* for I:

$$\hat{I} := \frac{f(x)}{p(x)} \tag{79}$$

A *secondary estimator* is obtained by averaging several primary estimators:

$$\hat{I}_N := \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{80}$$

where $(x_i)_{i \in \{1,\dots N\}}$ is a sequence of independent samples distributed according to p. The laws of large numbers states that $\hat{I}_N$ converges almost surely to I as $N \to \infty$.

3.2.2 *Estimation of a sum*

MC estimation can also be used to estimate a sum using a discrete probability distribution. Let S be the sum of terms $(a_i)_{i \in [\![1,k]\!]}$:

$$S = \sum_{i=1}^{k} a_i \tag{81}$$

Let X be a discrete random variable on $[\![1, k]\!]$ distributed according to a pmf p. The sum can be rewritten:

$$S = \sum_{i=1}^{k} \frac{a_i}{p(i)} p(i) \tag{82}$$

showing that $I = \mathbb{E}[\frac{a_X}{p(X)}]$. If $(n_i)_{i \in \{1,\dots N\}}$ is a sequence of independent samples distributed according to p, the secondary estimator for S is given by:

$$\hat{S}_N := \frac{1}{N} \sum_{i=1}^{N} \frac{a_{n_i}}{p(n_i)} \tag{83}$$

Note that this estimation can also be applied on infinite sums using a well defined pmf.

3.3 VARIANCE REDUCTION TECHNIQUES

In this section, we derive an expression for the variance of a MC estimator, which is in relation with the error produced by the estimation. We then describe variance reduction techniques we employed for the works presented in this thesis. We refer the reader to Veach's PhD thesis [Vea97, Chapter 2] and the book "Physically based Rendering" from Pharr and Humphreys [PH10, Chapter 14] for a more comprehensive overview of variance reduction methods often used in MC rendering.

3.3.1 *Variance of a Monte Carlo estimator*

A MC estimator $\hat{I}_N$ is said to be *unbiased*, meaning that its expected value is equal to the quantity to estimate, in our case the value of the integral. An interesting property of unbiased

estimators is that the *expected mean squared error* $\text{MSE}(\hat{I}_N) := \mathbb{E}[(\hat{I}_N - I)^2]$ is equal to the variance $\mathbb{V}[\hat{I}_N]$ of the estimator.

The variance of the estimator is:

$$\mathbb{V}[\hat{I}_N] = \mathbb{V}[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}] = \frac{1}{N^2}\mathbb{V}[\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}] \tag{84}$$

Since the samples are independent, the variance is linear over the sum. Moreover, the samples are identically distributed, thus we have $\forall i \in [\![1, N]\!]$, $\mathbb{V}[\frac{f(x_i)}{p(x_i)}] = \mathbb{V}[\frac{f(X)}{p(X)}]$. We obtain:

$$\mathbb{V}[\hat{I}_N] = \frac{1}{N^2}\mathbb{V}[\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}] = \frac{N}{N^2}\mathbb{V}[\frac{f(X)}{p(X)}] = \frac{1}{N}\mathbb{V}[\frac{f(X)}{p(X)}] \tag{85}$$

If the variance $\mathbb{V}[\frac{f(X)}{p(X)}]$ is finite, it decreases at a rate $\mathcal{O}(N)$ and the absolute error decreases at a rate $\mathcal{O}(\sqrt{N})$. MC estimators have a low convergence rate since four more samples are needed to divide the error by two. In order to decrease the error without increasing the number of samples, the variance $\mathbb{V}[\frac{f(X)}{p(X)}]$ needs to be reduced as much as possible. This factor is expressed by:

$$\mathbb{V}[\frac{f(X)}{p(X)}] = \mathbb{E}[(\frac{f(X)}{p(X)})^2] - \mathbb{E}[\frac{f(X)}{p(X)}]^2 = \int_{x\in\Omega}(\frac{f(x)}{p(x)})^2 p(x)dx - I^2 = \int_{x\in\Omega}\frac{f(x)^2}{p(x)}dx - I^2 \tag{86}$$

We observe that the higher is the ratio $\frac{f(X)^2}{p(X)}$, the higher is the variance of the estimator. To obtain $\mathbb{V}[\frac{f(X)}{p(X)}] = 0$, the optimal pdf is $p_o(x) = f(x)/\int_\Omega f = f(x)/I$. Indeed:

$$\mathbb{V}[\frac{f(X)}{p_o(X)}] = \int_{x\in\Omega}\frac{f(x)^2}{\frac{f(x)}{I}}dx - I^2 = I.\int_{x\in\Omega}f(x)dx - I^2 = I^2 - I^2 = 0 \tag{87}$$

Unfortunately, computing this pdf requires the knowledge of I, the value we are estimating.

3.3.2 *Importance sampling*

As mentioned previously, the optimal pdf p_o to use for MC integration is the normalization of f ($p_o(x) = f(x)/\int_\Omega f$), but this pdf requires the knowledge of the integral we want to estimate. Importance sampling consists in choosing a pdf p that mimics the shape of f (high probability density when f is high and the opposite when f is low) in order to reduce the variance of MC estimation. Indeed, the variance of a MC estimator is:

$$\mathbb{V}[\hat{I}_N] = \frac{1}{N}(\int_{x\in\Omega}\frac{f(x)^2}{p(x)}dx - I^2) \tag{88}$$

For a fixed number of samples N, this expression varies like $\int_{x\in\Omega}\frac{f(x)^2}{p(x)}dx$. If the integrand $\frac{f(x)^2}{p(x)}$ is high at many points, the variance is high too. To compensate for high values of $f(x)^2$, a high probability density $p(x)$ must be assigned to x. In rendering, the variance introduced

by a bad choice of p often appears as "speckles" (also know as "fireflies"), which are bright spots on the image that traduce samples chosen with low probability density but having high contribution.

Choosing an adapted pdf requires some prior information about the function to integrate. A simple example is the integral expressing the reflected radiance at a point:

$$L_s(\mathbf{x}, \omega_o) = \int_{\omega_i \in \mathcal{S}^2} L_i(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o)(n_\mathbf{x} \cdot \omega_i) d\sigma(\omega_i) \tag{89}$$

The integrand is a product of the incident radiance, the BSDF and the cosine factor (expressed as a dot product). The BSDF and the cosine factor are local components while the incident radiance is a global component that has no analytical expression and must also be estimated. When the BSDF has a simple form, it is generally possible to sample directions according to a pdf proportional to the product of the BSDF and the cosine, so that this factor is importance sampled. This strategy works well as long as the incident radiance does not expose high variations, which is the case for most outdoor scenes. However, for indoor scenes containing many occlusions, avoiding importance sampling of the incident radiance can lead to high variance traduced by strong noise in rendered images.

3.3.3 *Multiple importance sampling*

Multiple importance sampling (MIS) [VG95] is a general method to combine multiple sampling strategies to estimate an integral. The motivation behind MIS is that we are often able to importance sample individually each factor f_i of a function $f = \prod_{i=1}^{k} f_i$, but not the whole product. By combining in an optimal way all the sampling strategies, we can obtain a low variance estimator.

Let $(p_i)_{i \in [\![1,k]\!]}$ be a sequence of pdfs defined over a domain Ω on which we want to integrate f. The *multi-sample* MIS estimator is defined by:

$$\hat{I}_N^{\text{MIS}} := \sum_{i=1}^{k} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})} \tag{90}$$

where $(x_{i,j})_{j \in [\![1,N_i]\!]}$ is a sequence of samples distributed according to $X_i \sim p_i$, such that $N = \sum_{i=1}^{k} N_i$. A weighting function w_j is associated to each sampling strategy p_j to ensure that the estimator is unbiased. Indeed, the contribution of an element $x \in \Omega$ must be weighted to take into account that it can be sampled by multiple sampling strategies. The weighting functions must respect two conditions:

$$\forall x \in \Omega, \sum_{i=1}^{k} w_i(x) = 1 \tag{91}$$

$$p_i(y) = 0 \Rightarrow w_i(y) = 0 \tag{92}$$

Under these conditions, it is easy to prove that the estimator is unbiased:

$$\begin{aligned}
\mathbb{E}[\hat{I}_N] &= \mathbb{E}[\sum_{i=1}^{k} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})}] \\
&= \sum_{i=1}^{k} \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbb{E}[w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})}] \\
&= \sum_{i=1}^{k} \frac{N_i}{N_i} \mathbb{E}[w_i(X_i) \frac{f(X_i)}{p_i(X_i)}] \\
&= \sum_{i=1}^{k} \int_{x \in \Omega} w_i(x) \frac{f(x)}{p_i(x)} p_i(x) dx \\
&= \int_{x \in \Omega} (\sum_{i=1}^{k} w_i(x)) f(x) dx \\
&= \int_{x \in \Omega} f(x) dx
\end{aligned} \tag{93}$$

The standard MC estimator can be seen as a special case of the multi-sample MIS MC estimator where $k = 1$.

To compute a multi-sample MIS estimate, at least one sample must be drawn from each strategy, which is sometimes not practical. The *one-sample* MIS estimator can be used to take advantage of MIS with only one sample, obtained from a randomly chosen strategy:

$$\hat{I}^{\text{MIS}} := w_i(x_i) \frac{f(x_i)}{p_I(i).p_i(x_i)} \tag{94}$$

where $i \in [\![1, k]\!]$ is a realization of a discrete random variable I distributed according to the pmf p_I ($p_I(i)$ is the probability of using the strategy p_i) and x_i is a realization of the random variable $X_i \sim p_i$.

The weighting functions have to be chosen to reduce the variance of the estimator. Veach and Guibas [VG95] proposed the *power heuristic*, a family of weighting functions that depend on a parameter α:

$$\forall x \in \Omega, w_i(x) = \frac{(c_i.p_i(x))^\alpha}{\sum_{j=1}^{k} (c_j.p_j(x))^\alpha} \tag{95}$$

where $c_i = p_I(i)$ for the one-sample model and $c_i = N_i$ for the multi-sample model.

The choice $\alpha = 1$ is referred as the *balance heuristic* and Veach and Guibas shown that it is optimal for the one-sample estimator and nearly optimal for the multi-sample estimator. The choice $\alpha = 0$ corresponds to a constant weight $1/k$ given to each sample. The choice $\alpha = +\infty$ is referred as the *max heuristic* and gives a weight $w_i(x) > 0$ if and only if p_i is the strategy that gives the highest probability density to x.

Multiple importance sampling is an essential component of the bidirectional path tracing algorithm that allows to combine many sampling strategies for each sampled path while reducing dramatically the variance of the estimator when the balance heuristic is used.

3.3.4 *Resampling*

Suppose given the following MC estimator:

$$\hat{I}_N := \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{96}$$

such that each term $f(x_i)/p(x_i)$ is expensive to evaluate (if a ray has to be traced, for example) and p does not importance sample well enough the function f (the variance of the estimator is high). In that case, avoiding the evaluation of all the terms can increase the efficiency of the algorithm. Usually, new information can be extracted from the sequence of samples $(x_i)_{i\in[\![1,N]\!]}$ in order to define a resampling pmf $p_R : [\![1,N]\!] \to [0,1]$ that gives to each sample a probability of being selected according to an approximation of its contribution. Resampling applies MC estimation to the sum $\hat{I}_N$ using the pmf p_R:

$$\hat{I}^{\mathrm{R}}_{N,M} := \frac{1}{M}\sum_{j=1}^{M}\frac{f(x_{i_j})}{N.p(x_{i_j}).p_R(i_j)} \tag{97}$$

where $(i_j)_{j\in[\![1,M]\!]}$ is a sequence of index distributed according to the pmf p_R. This estimator is unbiased as long as $f(x_{i_j}) \neq 0 \Rightarrow p_R(i_j) > 0$.

Note that this estimator is equivalent to the *resampled importance sampling* (RIS) estimator proposed by Talbot et al.[TCE05] and defined by:

$$\hat{I}^{\mathrm{RIS}}_{N,M} := \frac{1}{M}\Big(\sum_{j=1}^{M}\frac{f(x_{i_j})}{\tilde{q}(x_{i_j})}\Big)\Big(\frac{1}{N}\sum_{k=1}^{N}w(x_k)\Big) \tag{98}$$

where $\tilde{q}$ represents an unormalized pdf from which we want to sample elements, w is a weighting function defined by $w(x) = \tilde{q}(x)/p(x)$ and $(i_j)_{j\in[\![1,M]\!]}$ is a sequence of index distributed according to the pmf p_R defined by $p_R(i) = w(x_i)/\sum_{k=1}^{N}w(x_k)$. Replacing $p_R(i_j)$ with this definition in Equation (97), we obtain:

$$\begin{aligned}
\hat{I}^{\mathrm{R}}_{N,M} &= \frac{1}{M}\sum_{j=1}^{M}\frac{f(x_{i_j})}{N.p(x_{i_j}).\frac{w(x_{i_j})}{\sum_{k=1}^{N}w(x_k)}} \\
&= \frac{1}{M}\sum_{j=1}^{M}\Big(\frac{f(x_{i_j})}{p(x_{i_j}).w(x_{i_j})}\Big)\Big(\frac{1}{N}\sum_{k=1}^{N}w(x_k)\Big) \\
&= \frac{1}{M}\Big(\sum_{j=1}^{M}\frac{f(x_{i_j})}{\tilde{q}(x_{i_j})}\Big)\Big(\frac{1}{N}\sum_{k=1}^{N}w(x_k)\Big) \\
&= \hat{I}^{\mathrm{RIS}}_{N,M}
\end{aligned} \tag{99}$$

Resampling is well adapted when $\tilde{q}$ cannot be sampled directly but importance samples the contribution function f better than the sampling pdf p.

3.3.5 *Russian roulette*

Russian roulette, introduced by Arvo et al. [AK90] in light transport simulation, is a method to avoid evaluating an estimator based on a stopping probability $p_{\mathrm{stop}} \in [0,1)$. It is mostly used

to stop the iterative sampling of a path without biasing the estimator. Let $\hat{Q}$ be an unbiased estimator for some quantity Q. The Russian roulette estimator $\hat{Q}_{\text{RR}}(p_{\text{stop}})$ is defined by:

$$\hat{Q}_{\text{RR}}(p_{\text{stop}}) = \begin{cases} 0 & \text{with probability } p_{\text{stop}} \\ \frac{\hat{Q}}{1-p_{\text{stop}}} & \text{with probability } 1 - p_{\text{stop}} \end{cases} \tag{100}$$

This estimator is unbiased since:

$$\mathbb{E}[\hat{Q}_{\text{RR}}(p_{\text{stop}})] = p_{\text{stop}}.0 + (1 - p_{\text{stop}}).\frac{\mathbb{E}[\hat{Q}]}{1 - p_{\text{stop}}} = \mathbb{E}[\hat{Q}] = Q \tag{101}$$

4

PATH SAMPLING RENDERING ALGORITHMS

Nowadays, most of the state-of-the-art methods to estimate light transport equations are based on path sampling solutions. This chapter provides an overview of standard algorithms of the path sampling framework. We first describe local path sampling in Section 4.1, that is used by all algorithms presented in this chapter. Path tracing, bidirectional path tracing and many-light rendering rely on Monte-Carlo integration and are respectively detailed in Section 4.2, 4.3, 4.4. Photon mapping, described in Section 4.5, uses density estimation on path vertices to estimate radiance. Finally, Section 4.6 gives a brief overview of Markov chain Monte Carlo rendering, another common alternative that samples paths according to their contribution on the image, using random mutations.

Contents

4.1 LOCAL PATH SAMPLING

In order to estimate the measurement equation in the path space framework (Equation (38)) with Monte Carlo integration, path sampling strategies must be defined. Recall that the contribution of a path to a measurement I_i is given by:

$$\begin{aligned} f^{(i)}(\mathbf{x}_1 ... \mathbf{x}_k) &= L_e(\mathbf{x}_1, \mathbf{x}_2) T(\mathbf{x}_1 ... \mathbf{x}_k) W_e^{(i)}(\mathbf{x}_k, \mathbf{x}_{k-1}) \\ T(\mathbf{x}_1 ... \mathbf{x}_k) &= \left[\prod_{i=1}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) \prod_{i=2}^{k-1} f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \right] \\ I_i &= \int_{\bar{\mathbf{x}} \in \mathcal{P}} f^{(i)}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \end{aligned} \tag{102}$$

For maximal efficiency, path sampling strategies must importance sample all factors of the path contribution function while being computationally cheap to evaluate. This problem is hard to solve due to the infinite dimensionality of the path space and the discontinuities in the contribution function introduced by visibility factors and specular scattering. In this section, we first introduce the concepts of *eye sub-paths* and *light sub-paths*. Then we review two important path sampling strategies based on these concepts: unidirectional path sampling and bidirectional path sampling.

4.1.1 *Light sub-paths and eye-sub paths*

A *light sub-path* $\bar{\mathbf{y}}_s = \mathbf{y}_1 ... \mathbf{y}_s$, with $s \geq 0$, is a path starting at a light source for which any vertex is visible from its successor:

$$\begin{aligned} \mathbf{y}_1 &\in \mathcal{M}_L \\ \forall i \in [\![1, s-1]\!], \quad V(\mathbf{y}_i, \mathbf{y}_{i+1}) &= 1 \end{aligned} \tag{103}$$

When $s = 0$, the light sub-path is the *empty light sub-path* $\bar{\mathbf{y}}_0$. We denote $\mathcal{P}^L$ the space of light sub-paths. We refer to vertices from a light sub-path as *light vertices*. Figure 12 illustrates a light sub-path.

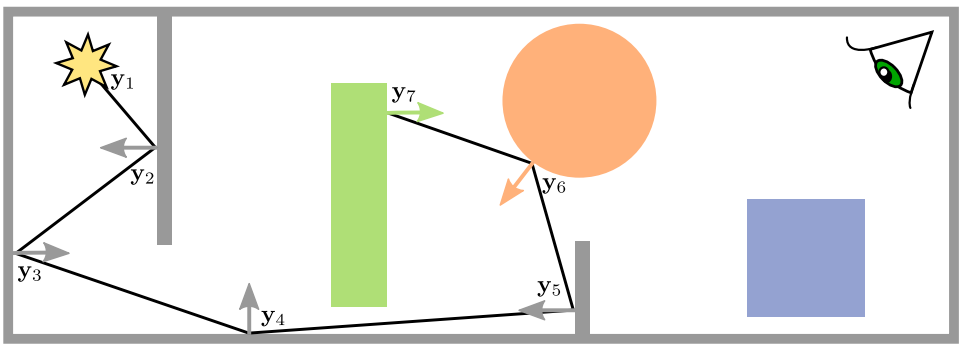


Figure 12 – Illustration of a light sub-path.

Similarly, an *eye-sub path* $\bar{\mathbf{z}}_t = \mathbf{z}_1 ... \mathbf{z}_t$, with $t \geq 0$, is a path starting at a sensor, satisfying the conditions:

$$\begin{aligned} \mathbf{z}_1 &\in \mathcal{M}_W \\ \forall i \in [\![1, t-1]\!], \quad V(\mathbf{z}_i, \mathbf{z}_{i+1}) &= 1 \end{aligned} \tag{104}$$

When $t = 0$, the eye sub-path is the *empty eye sub-path* $\bar{\mathbf{z}}_0$. We denote $\mathcal{P}^E$ the space of eye sub-paths. We refer to vertices from an eye sub-path as *eye vertices*. Figure 13 illustrates an eye sub-path.

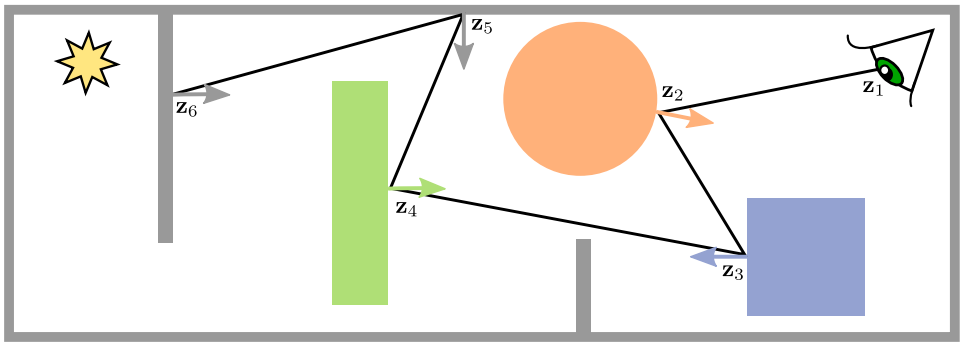


Figure 13 – Illustration of an eye sub-path.

4.1.2 *Unidirectional path sampling*

Unidirectional path sampling performs a random walk in the scene, starting either at a sensor i by sampling a vertex distributed proportionally to the sensor response $W_e^{(i)}$, or at a light source by sampling a vertex distributed proportionally to the emitted radiance L_e. Supposing a sub-path $\mathbf{x}_1 ... \mathbf{x}_k$ as already been obtained, the next vertex $\mathbf{x}_{k+1}$ is sampled conditionally to the current vertex $\mathbf{x}_k$, distributed proportionally to the BSDF and the geometric factor. To achieve this, we suppose that the following pdfs are available and can be sampled with numerical methods (see Figure 14 for an illustration):

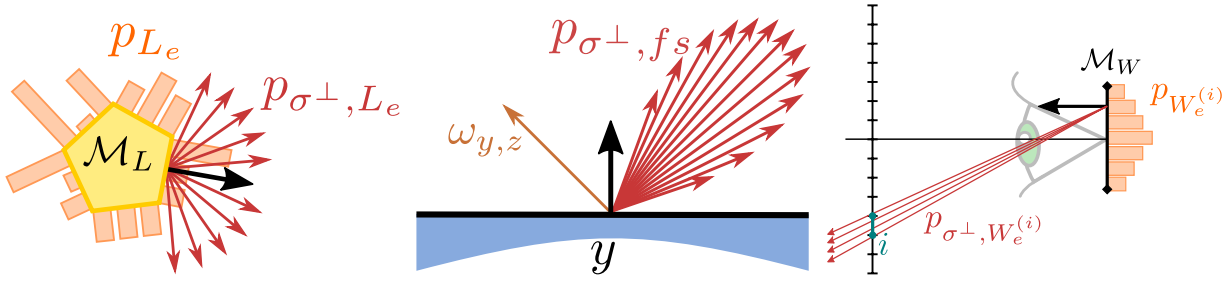
— $p_{W_e^{(i)}}$ samples surface points proportionally to their sensor response, with respect to area measure.

— $p_{\sigma^\perp, W_e^{(i)}}$ samples directions ω proportionally to $W_e^{(i)}(\mathbf{x}, \omega)$ for a surface point $\mathbf{x}$. Given $\mathbf{y} \in \mathcal{M}$, we note $p_{\sigma^\perp, W_e^{(i)}}(\mathbf{x} \to \mathbf{y})$ the probability density of $\omega_{\mathbf{x},\mathbf{y}}$ with respect to projected solid angle measure at $\mathbf{x}$.

— p_{L_e} samples surface points proportionally to their emitted radiance, with respect to area measure.

— $p_{\sigma^\perp, L_e}$ samples directions ω proportionally to $L_e(\mathbf{x}, \omega)$ for a surface point $\mathbf{x}$. Given $\mathbf{y} \in \mathcal{M}$, we note $p_{\sigma^\perp, L_e}(\mathbf{x} \to \mathbf{y})$ the probability density of $\omega_{\mathbf{x},\mathbf{y}}$ with respect to projected solid angle measure at $\mathbf{x}$.

— $p_{\sigma^\perp, f_s}$ samples directions ω proportionally to $f_s(\mathbf{y}, \omega_{\mathbf{y},\mathbf{x}}, \omega)$ for surface points $\mathbf{x}$, $\mathbf{y}$. Given $\mathbf{z} \in \mathcal{M}$, we note $p_{\sigma^\perp, f_s}(\mathbf{x} \to \mathbf{y} \to \mathbf{z})$ the probability density of $\omega_{\mathbf{y},\mathbf{z}}$ with respect to projected solid angle measure at $\mathbf{y}$.



(a) light source sampling (b) BSDF sampling (c) sensor sampling

Figure 14 – Illustration of pdfs to sample directions, and surface points on light and sensor surfaces. Surface sampling pdfs are illustrated as probability histograms. Note that these ones are just fictive examples, as they are most often just set to a uniform probability with respect to area, or computed from illumination textures in the case of light sources.

A probability density $p_{\sigma^\perp}(\omega)$ expressed with respect to projected solid angle at a point $\mathbf{x}$ can be converted to a probability density expressed with respect to area, using the identity:

$$p_{\mathcal{A}}(\mathbf{y}) = G(\mathbf{x},\mathbf{y})p_{\sigma^\perp}(\omega_{\mathbf{x},\mathbf{y}}) \tag{105}$$

This pdf naturally importance samples the geometric factor, so if $p_{\sigma^\perp}(\omega)$ importance samples the directional factor at $\mathbf{x}$ (L_e on light sources, W_e on sensors and fs on surfaces), then $p_{\mathcal{A}}$ importance samples the product of the two factors. We note $p_{\mathcal{A},W_e^{(i)}}$, $p_{\mathcal{A},L_e}$ and $p_{\mathcal{A},f_s}$ the pdfs expressed with respect to area corresponding to $p_{\sigma^\perp,W_e^{(i)}}$, $p_{\sigma^\perp,L_e}$ and $p_{\sigma^\perp,f_s}$, respectively.

An eye sub-path $\bar{\mathbf{z}}_t = \mathbf{z}_1...\mathbf{z}_t$, $\mathbf{z}_1 \in \mathcal{M}_W^{(i)}$ of length t starting at sensor i can then be sampled iteratively according to the path pdf $p_{E,t}^{(i)}$ defined by:

$$p_{E,t}^{(i)}(\mathbf{z}_1...\mathbf{z}_t) = p_{W_e^{(i)}}(\mathbf{z}_1)p_{\mathcal{A},W_e^{(i)}}(\mathbf{z}_1 \to \mathbf{z}_2)\prod_{i=2}^{t-1} p_{\mathcal{A},f_s}(\mathbf{z}_{i-1} \to \mathbf{z}_i \to \mathbf{z}_{i+1}) \tag{106}$$

The path $\bar{\mathbf{z}}_t$ contributes to the measurement $I_{i,t}$ (Equation (43)) only if $L_e(\mathbf{z}_t,\mathbf{z}_{t-1}) > 0$. Consequently, the pdf $p_{E,t}^{(i)}$ importance samples all factors of the contribution function except the emitted radiance at the last vertex, making it inefficient for small light sources or light sources accessible by narrow openings, such as small windows in an house.

The strategy $p_{E,t}^{(i)}$ does not sample the whole path space $\mathcal{P}$ but only paths of length t. However, each path of length t can be seen as $t-1$ paths of lengths t, $t-1$, ..., 2 that can be used to estimates the terms $I_{i,t}$, $I_{i,t-1}$, ..., $I_{i,2}$ of the measurement equation. Estimating only a fixed number of terms gives a biased estimator. Russian roulette can be used to decide when to stop the random walk instead of setting a maximal path length. At each vertex $\mathbf{z}_i$, $i \geq 2$ of a path, we define a probability $p_{rr}(\mathbf{z}_i)$ of extending the path. This probability is generally computed from the material properties of the vertex given its incident direction (the more the vertex absorbs energy, the lower is the probability of extending the path).

For an eye path $\bar{\mathbf{z}}_t$, we define:

$$p_{\bar{\mathbf{z}}_t}(\mathbf{z}_i) := \begin{cases} p_{W_e^{(i)}}(\mathbf{z}_1) & \text{if } i = 1 \\ p_{\mathcal{A},W_e^{(i)}}(\mathbf{z}_1 \to \mathbf{z}_2) & \text{if } i = 2 \\ p_{\text{rr}}(\mathbf{z}_i).p_{\mathcal{A},f_s}(\mathbf{z}_{i-2} \to \mathbf{z}_{i-1} \to \mathbf{z}_i) & \text{otherwise.} \end{cases} \tag{107}$$

and we set:

$$\begin{aligned} p_E^{(i)}(\bar{\mathbf{z}}_t) : &= \prod_{i=1}^{t} p_{\bar{\mathbf{z}}_t}(\mathbf{z}_i) \\ &= p_E^{(i)}(\bar{\mathbf{z}}_{t-1}) p_{\bar{\mathbf{z}}_t}(\mathbf{z}_t) \end{aligned} \tag{108}$$

Similarly, a light sub-path $\bar{\mathbf{y}}_s = \mathbf{y}_1 ... \mathbf{y}_s$, can be sampled with the path pdf $p_{L,s}$ defined by:

$$p_{L,s}(\mathbf{y}_1 ... \mathbf{y}_s) = p_{L_e}(\mathbf{y}_1) p_{\mathcal{A},L_e}(\mathbf{y}_1 \to \mathbf{y}_2) \prod_{i=2}^{s-1} p_{\mathcal{A},f_s}(\mathbf{y}_{i-1} \to \mathbf{y}_i \to \mathbf{y}_{i+1}) \tag{109}$$

The only factor not importance sampled by $p_{L,s}$ is the emitted importance $W_e^{(i)}(\mathbf{y}_s, \mathbf{y}_{s-1})$, which is more problematic than not importance sampling L_e since the sensor surface is generally represented by a unique and small camera lens, thus the probability of reaching it is extremely low.

For a light path $\bar{\mathbf{y}}_s$, we define:

$$p_{\bar{\mathbf{y}}_s}(\mathbf{y}_i) := \begin{cases} p_{L_e}(\mathbf{y}_1) & \text{if } i = 1 \\ p_{\mathcal{A},L_e}(\mathbf{y}_1 \to \mathbf{y}_2) & \text{if } i = 2 \\ p_{\text{rr}}(\mathbf{y}_i).p_{\mathcal{A},f_s}(\mathbf{y}_{i-2} \to \mathbf{y}_{i-1} \to \mathbf{y}_i) & \text{otherwise.} \end{cases} \tag{110}$$

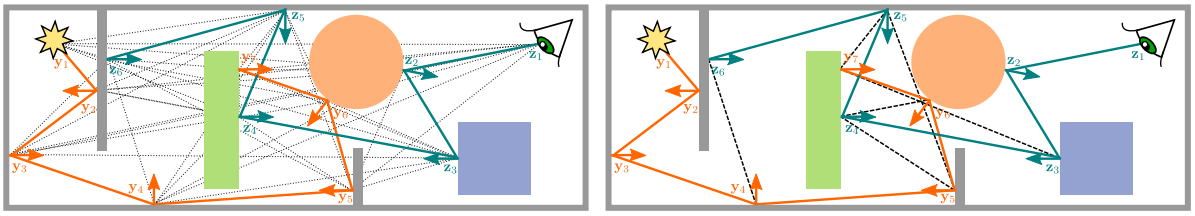
and we set:

$$\begin{aligned} p_L(\bar{\mathbf{y}}_s) : &= \prod_{i=1}^{s} p_{\bar{\mathbf{y}}_s}(\mathbf{y}_i) \\ &= p_L(\bar{\mathbf{y}}_{s-1}) p_{\bar{\mathbf{y}}_s}(\mathbf{y}_s) \end{aligned} \tag{111}$$

4.1.3 *Bidirectional path sampling*

The sampling strategy $p_E^{(i)}$ (resp. p_L) does not importance sample the emitted radiance (resp. emitted importance) factor. When this factor dominates the contribution of a path, and the probability of sampling that path is low, the associated Monte Carlo estimator has a high variance and, therefore, slow convergence.

Bidirectional path sampling exploits the strengths of both strategies $p_E^{(i)}$ and p_L by creating a collection of path sampling strategies $p_{s,t}$ for each path length k. Each strategy $p_{s,t}$ connects an eye sub-path of length t with a light sub-path of length s such that $s + t = k$. The number of strategies created that way is $k + 1$, the number of pairs $(s, t) \in \mathbb{N}^2$ such that $s + t = k$.



(a) All connections (b) Unoccluded connections

Figure 15 – Illustration of a sequence of complete paths sampled by bidirectional sampling strategies. Only few of these paths actually contribute to the image because of occlusions.

Let $\bar{\mathbf{y}}_S = \mathbf{y}_1...\mathbf{y}_S$ be a light sub-path obtained with sampling strategy p_L and $\bar{\mathbf{z}}_T = \mathbf{z}_1...\mathbf{y}_T$ be an eye sub-path obtained with sampling strategy $p_E^{(i)}$. A sequence of correlated paths $(\bar{\mathbf{x}}_{s,t} = \mathbf{y}_1...\mathbf{y}_s\mathbf{z}_t...\mathbf{z}_1)_{2\leq s+t\leq S+T}$ is obtained by connecting each eye vertex with each light vertex. Each path $\bar{\mathbf{x}}_{s,t}$ is distributed according to the sampling strategy $p_{s,t}$ defined by:

$$p_{s,t}(\bar{\mathbf{x}}_{s,t}) = \begin{cases} p_L(\bar{\mathbf{y}}_s) & \text{if t = o} \\ p_E^{(i)}(\bar{\mathbf{z}}_t) & \text{if s = o} \\ p_L(\bar{\mathbf{y}}_s).p_E^{(i)}(\bar{\mathbf{z}}_t) & \text{otherwise.} \end{cases} \tag{112}$$

The contribution $f^{(i)}(\bar{\mathbf{x}}_{s,t})$ of a complete path can be expressed as the product of a partial contribution $f_E(\bar{\mathbf{z}}_t)$ from the eye sub-path, with a partial contribution $f_L(\bar{\mathbf{y}}_s)$ from the light sub-path, with an additional connection factor $C(\bar{\mathbf{x}}_{s,t})$:

$$f^{(i)}(\bar{\mathbf{x}}_{s,t}) = f_L(\bar{\mathbf{y}}_s).C(\bar{\mathbf{x}}_{s,t}).f_E^{(i)}(\bar{\mathbf{z}}_t) \tag{113}$$

$$f_L(\bar{\mathbf{y}}_s) = \begin{cases} 1 & \text{if } s \leq 1 \\ L_e(\mathbf{y}_1, \mathbf{y}_2)T(\mathbf{y}_1...\mathbf{y}_s) & \text{otherwise.} \end{cases} \tag{114}$$

$$f_E(\bar{\mathbf{z}}_t) = \begin{cases} 1 & \text{if } t \leq 1 \\ W_e^{(i)}(\mathbf{z}_1, \mathbf{z}_2)T(\mathbf{z}_1...\mathbf{z}_t) & \text{otherwise.} \end{cases} \tag{115}$$

The connection factor $C(\bar{\mathbf{x}}_{s,t})$ contains the missing factors from the contribution:

$$C(\bar{\mathbf{x}}_{s,t}) = \begin{cases} L_e(\mathbf{z}_t, \mathbf{z}_{t-1}) & \text{if } s = 0 \\ L_e(\mathbf{y}_1, \mathbf{z}_t)G(\mathbf{y}_1, \mathbf{z}_t) & \text{if } s = 1 \\ W_e^{(i)}(\mathbf{y}_s, \mathbf{y}_{s-1}) & \text{if } t = 0 \\ W_e^{(i)}(\mathbf{z}_1, \mathbf{y}_s)G(\mathbf{z}_1, \mathbf{y}_s) & \text{if } t = 1 \\ f_s(\mathbf{y}_{s-1}, \mathbf{y}_s, \mathbf{z}_t)G(\mathbf{y}_s, \mathbf{z}_t)f_s(\mathbf{y}_s, \mathbf{z}_t, \mathbf{z}_{t-1}) & \text{otherwise.} \end{cases} \tag{116}$$

Similarly to $p_E^{(i)}$ and p_L, the sampling strategy $p_{s,t}$ importance samples all factors of the contribution function except $C(\bar{\mathbf{x}}_{s,t})$. When this factor is low (potentially null when an occlusion occurs between the connection vertices) for most paths sampled with high probability by the strategy, the variance of the estimator associated with $p_{s,t}$ is high (Figure 15). All of these strategies works well for a subset of paths and choosing the best strategy depending on the current local configuration is still unresolved. Nevertheless, it remains possible to combine all the strategies with multiple importance sampling in the context of bidirectional path tracing (detailed in Section 4.3).

4.1.4 *Improving ray sampling with density estimation*

Ray sampling can be improved by working on the probability density function used to sample a new ray at each vertex. Indeed, only sampling according to the BSDF can result in a poor estimator in scenes where the illumination is highly indirect. To improve that sampling, global information must be used concerning the distribution of radiance or importance in the scene. A common solution is to trace particles from the light sources or from the sensors and to guide the sampling of rays according to the distribution of these particles. This strategy of reconstructing a sampling pdf from a distribution of particles is a density estimation problem.

One of the first method built on this approach was introduced by Jensen [Jen95] in order to sample rays according to incident radiance. For that, a ray sampling pdf is reconstructed at each surface point $\mathbf{x}$ of a path using histogram density estimation from a distribution of photons stored in a precomputed photon map. The hemisphere $\mathcal{S}^+(\mathbf{x})$ of the point is partitioned into bins of constant size, and each bins gets assigned a probability that depends on the number of nearest photons having their incident direction falling in the bin. A outgoing direction is then sampled according to this histogram, using standard sampling of piecewise constant 2D distributions. Peter and Pietrek [PPI98] extended this idea to the sampling of rays according to sensor response, by pre-computing an importon map instead of a photon map (an *importon* is a particle emitted from a sensor).

This approach has three major problems: the cost associated to the construction and sampling of the estimated pdfs, the dependency between the density of photons available and the estimated pdf, and the constant size of bins that do not adapt to details in the incident radiance function. While the method could be made progressive, by caching the estimated pdfs and reusing them at nearest points, this does not solve the first and third problem.

Addressing the third problem, Hey and Purgathofer [HP02] replace the constant size bins with cones of adaptive width centered at gathered photons. While this strategy allows a more robust estimation of the sampling pdfs, it also induces a substantial overhead.

More recently, Vorba et al. [VKŠ+14] introduced an approach to learn pdfs at a sparse set of surface points from streams of particles, in a progressive manner. Instead of histogram density estimation, they adopt parametric mixture model estimation, where each sampling pdf is represented by a gaussian mixture model for which parameters are estimated from the distribution of particles. Reconstructed pdfs are cached in the scene adaptively in order to reuse them for nearby points. While the pdfs are refined over time, thanks to the progressive learning approach, a initial training phase is required to start the rendering from pdfs that represent well enough the distribution of radiance/importance. This training phase can take a substantial amount of time for large and complex scenes. Also, querying the cache for nearest estimated pdfs incurs a high overhead, from 26% (on path tracing) to 45% (on bidirectional path tracing) according to their experiments.

In this thesis, we propose alternative ray sampling strategies, based on geometric and topologic information extracted from a discrete representation of the empty space of the scene (see Chapter 9 and Chapter 10).

4.2 PATH TRACING

Introduced by Kajiya [Kaj86], together with the rendering equation, path tracing is a simple and elegant Monte Carlo method to estimate the measurement equation for each pixel of an image.

Even though this algorithm is quite old, movie industry has recently shift from rasterization solutions to path tracing [KFF+15] as demonstrated by new production renderers (addition of path tracing to RenderMan from Pixar, Arnold from Solid Angle, Hyperion from Disney, Shining from Ubisoft, Maxwell Renderer, etc.). This can be explained by the fact that computers have become powerful enough to compute path tracing in a reasonable amount of time on large and complex scenes designed for movies. Path tracing provides more realistic and physically-based images by simulating lighting effects that are hard to obtain with rasterization, such as color bleeding, caustics or soft shadows. The shift also introduces new interesting challenges, such as artistic control over the light transport simulation and addition of non physically based effects in the Monte Carlo path tracing framework.

In this section, we present the algorithm in its most basic form as well as common optimizations to reduce its variance.

4.2.1 *Basic algorithm*

Path tracing traces paths reversely from physical light propagation: from the camera to the light sources of the scene. The reason for this is that only paths arriving at the camera lens and passing through the image have a chance to contribute to the intensity of the pixels. Basically, path tracing samples the path space according the sampling strategies $(p_{0,t})_{t>1}$ introduced in Section 4.1.3 and estimates the measurement equation with Monte Carlo integration (Figure 16).

Algorithm 1 : Path Tracing Algorithm

Data : A scene $\mathcal{M}$, a camera C, an image resolution $W \times H$
Result : An image $I : [\![1, W]\!] \times [\![1, H]\!] \to \mathbb{R}$

for $i = 1 \to W \times H$ **do**
  $I_i \leftarrow 0$
  sample $\mathbf{z}_1 ... \mathbf{z}_t \sim p_E^{(i)}$; `// Sample an eye path, the length` t `is random`
  for $k = 2 \to t$ **do**
    $I_i \leftarrow I_i + \frac{f^{(i)}(\mathbf{z}_k ... \mathbf{z}_1)}{p_{0,k}(\mathbf{z}_k ... \mathbf{z}_1)}$; `// Estimate` $I_{i,k}$
return I;

Algorithm 1 illustrates a single iteration of path tracing, where one path $\bar{\mathbf{x}} = \mathbf{z}_t...\mathbf{z}_1 \sim p_{0,t}$ is sampled and used to compute the following estimation of the measurement equation (35):

$$\hat{I}_i^{\mathrm{PT}} = \underbrace{\frac{f^{(i)}(\mathbf{z}_2\mathbf{z}_1)}{p_{0,2}(\mathbf{z}_2\mathbf{z}_1)}}_{\text{estimates } I_{i,2}} + \underbrace{\frac{f^{(i)}(\mathbf{z}_3\mathbf{z}_2\mathbf{z}_1)}{p_{0,3}(\mathbf{z}_3\mathbf{z}_2\mathbf{z}_1)}}_{\text{estimates } I_{i,3}} + ... + \underbrace{\frac{f^{(i)}(\mathbf{z}_t...\mathbf{z}_1)}{p_{0,t}(\mathbf{z}_t...\mathbf{z}_1)}}_{\text{estimates } I_{i,t}} \tag{117}$$

The other terms $(I_{i,k})_{k>t}$ are canceled by Russian roulette, thus their sum is estimated by zero. An estimation with N samples per pixel can be obtained by running N times the algorithm and averaging the resulting images. This technique is known as *progressive rendering* and is often used for interactive rendering when each iteration can be computed in real-time.

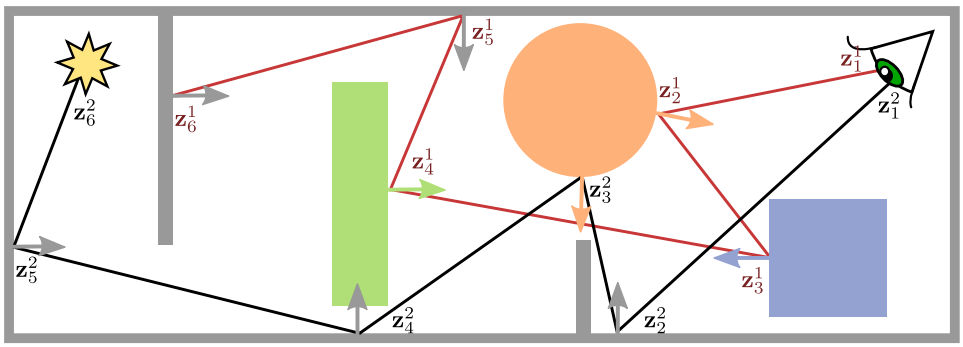


Figure 16 – Illustration of two paths sampled with path tracing. The path drawn in red does not reach the light source before termination, and thus has a null contribution.

4.2.2 *Variance*

With path tracing, variance of the Monte Carlo estimator appears as white noise in rendered images, due to independent random sampling for neighbor pixels. There is generally two kind of noise in path traced images:

— dark pixels, appearing when the path sampling strategy does not reach often enough the light source.

— bright pixels, referred as *speckles* or *fireflies*, appearing when the path sampling strategy exceptionally draws a high contributing path with low probability.

The first kind of noise generally implies the second, since it traduces a high probability given to low contribution paths, and a low probability given to high contribution paths. The second can also occurs in presence of highly directional materials, such as specular ones, since the probability of reaching the area of a specular material that bounce to a bright area of the scene is generally low when tracing paths from the eye.

4.2.3 *Next-event estimation*

As discussed in Section 4.1.3, using the strategies $(p_{0,t})_{t>1}$ can be quite inefficient in scenes containing small light sources. Even worse, the contribution of point and directional light sources cannot be simulated with this strategy since the probability of sampling the last

vertex of an eye path on such artificial light sources is null. Such light sources are not physically based but are often used to approximate distant or small sources. A solution to this problem is to sample explicitly the last vertex of the path on a light source, that is, using the sampling strategies $(p_{1,t})_{t>0}$ (Figure 17). In the context of path tracing, this optimization is referred as *next-event estimation*.

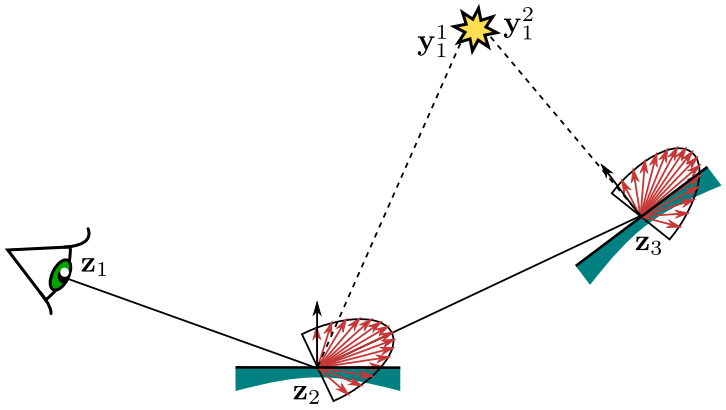


Figure 17 – Illustration of next-event estimation. Since surfaces are glossy in this example, the direction to the light source at vertices $\mathbf{z}_1$ and $\mathbf{z}_2$ is very unlikely to be sampled using the BSDF pdf (represented by red arrows). Sampling a vertex on the light source and connecting it to each eye vertex gives a higher probability of generating a contributing path.

4.2.4 *Multiple importance sampling*

The best sampling strategy to select the first vertex $\mathbf{x}_1 \in \mathcal{M}_L$ of a a contributing path $\mathbf{x}_1 ... \mathbf{x}_k$ depends on the lighting configuration and the BSDF at vertex $\mathbf{x}_2$. Figure 18 illustrates the strength of each sampling strategy on a simple scene. The scene is a reproduction, provided with the Mitsuba Renderer [Jak10], of the famous one introduced by Veach to illustrate MIS.

The strategy $p_{0,k}$, which sample the BSDF at $\mathbf{x}_2$ to obtain $\mathbf{x}_1$, works best if the edge $\mathbf{x}_2\mathbf{x}_1$ has a high contribution for the BSDF and the light source is quite large. On the opposite, if the BSDF does not give a high contribution to the edge and the light source is small, the strategy $p_{1,t-1}$ is better.

Without any prior information on the scene geometry around $\mathbf{x}_2$, we cannot guess the most adapted strategy without actually sampling the path. In order to improve the estimation in the case of a bad choice for the strategy, we can use multiple importance sampling, described in Section 3.3.3.

Let $\bar{\mathbf{x}}_{0,k} = \mathbf{z}_k ... \mathbf{z}_1$ be a path sampled with strategy $p_{0,k}$. Let $\mathbf{y}_1$ be a vertex sampled according to p_{L_e} and $\bar{\mathbf{x}}_{1,k-1} = \mathbf{y}_1 \mathbf{z}_{k-1} ... \mathbf{z}_k$ be the path obtained by connecting the light vertex $\mathbf{y}_1$ with the eye vertex $\mathbf{z}_{k-1}$, i.e. a path sampled with strategy $p_{1,k-1}$. These two paths belongs to the sub path space $\mathcal{P}_k$ and can be used to estimate $I_{i,k}$ with MIS:

$$\hat{I}_{i,k}^{\text{PT, MIS}} = w_{0,k}(\bar{\mathbf{x}}_{0,k}) \frac{f^{(i)}(\bar{\mathbf{x}}_{0,k})}{p_{0,k}(\bar{\mathbf{x}}_{0,k})} + w_{1,k-1}(\bar{\mathbf{x}}_{1,k-1}) \frac{f^{(i)}(\bar{\mathbf{x}}_{1,k-1})}{p_{1,k-1}(\bar{\mathbf{x}}_{1,k-1})} \tag{118}$$

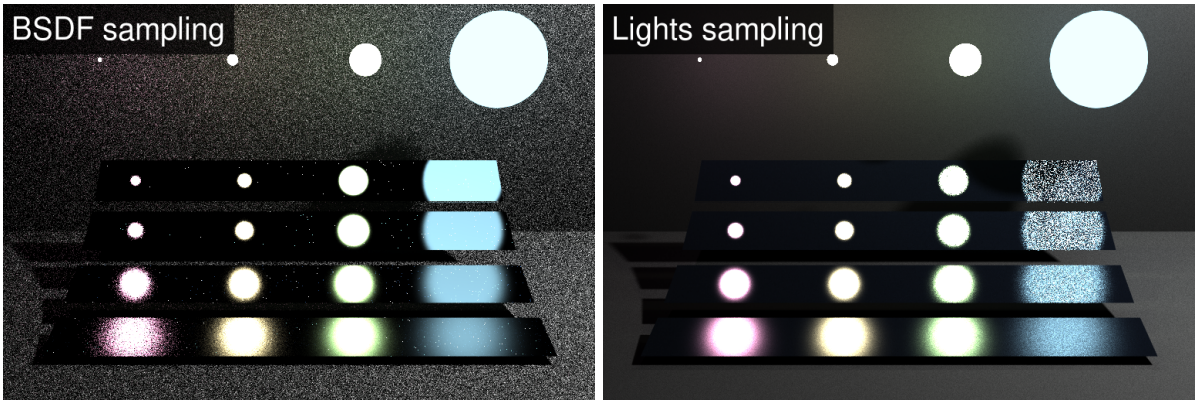


Figure 18 – A comparison of the strategy $p_{0,3}$ (left image) and the strategy $p_{1,2}$ (right image) after 30 seconds of rendering. The closest plate to the viewer is the roughest and the farthest plate is the most shiny.

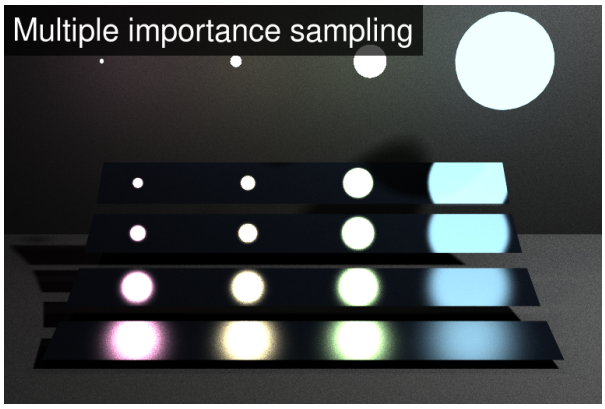


Figure 19 – Multiple importance sampling between strategies $p_{0,3}$ and $p_{1,2}$ (30 seconds of rendering). The strength of each sampling strategy is conserved such that the resulting estimator has low variance at every point.

With the balance heuristic, the weight $w_{0,k}(\bar{\mathbf{x}}_{0,k})$ is expressed by:

$$\begin{aligned} w_{0,k}(\bar{\mathbf{x}}_{0,k}) &= \frac{p_{0,k}(\bar{\mathbf{x}}_{0,k})}{p_{0,k}(\bar{\mathbf{x}}_{0,k}) + p_{1,k-1}(\bar{\mathbf{x}}_{0,k})} \\ &= \frac{p_{\bar{\mathbf{z}}}(\mathbf{z}_k).p_{0,k-1}(\bar{\mathbf{x}}_{0,k-1})}{p_{\bar{\mathbf{z}}}(\mathbf{z}_k).p_{0,k-1}(\bar{\mathbf{x}}_{0,k-1}) + p_{L_\mathrm{e}}(\mathbf{z}_k).p_{0,k-1}(\bar{\mathbf{x}}_{0,k-1})} \\ &= \frac{p_{\bar{\mathbf{z}}}(\mathbf{z}_k)}{p_{\bar{\mathbf{z}}}(\mathbf{z}_k) + p_{L_\mathrm{e}}(\mathbf{z}_k)} \end{aligned} \tag{119}$$

Similarly, the weight $w_{1,k-1}(\bar{\mathbf{x}}_{1,k-1})$ is expressed by:

$$w_{1,k-1}(\bar{\mathbf{x}}_{1,k-1}) = \frac{p_{L_\mathrm{e}}(\mathbf{y}_1)}{p_{\mathcal{A},f_\mathrm{s}}(\mathbf{z}_{k-1} \to \mathbf{y}_1) + p_{L_\mathrm{e}}(\mathbf{y}_1)} \tag{120}$$

These expressions demonstrate that MIS weights can be computed efficiently using local information available at vertices $\mathbf{x}_1$ and $\mathbf{x}_2$ of a path $\bar{\mathbf{x}}$. Figure 19 demonstrates the result of MIS applied to path tracing.

4.2.5 *Light tracing*

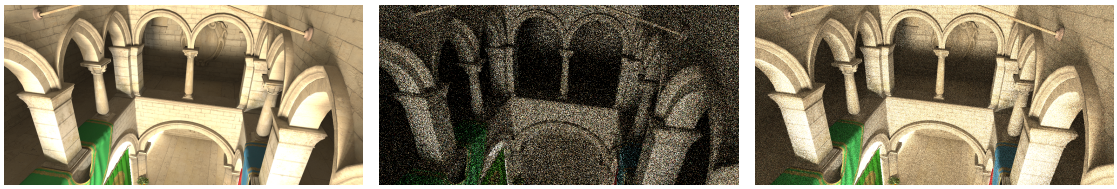
Path tracing can also be performed in the direction of light propagation and is referred as *light tracing* [DLW93]. Light tracing samples paths according to strategies $(p_{s,0})_{s>1}$, starting at a vertex on a light source and extending the path until it intersects the camera lens. Next-event estimation can be used by projecting path vertices directly on the camera lens, i.e. with sampling strategies $(p_{s-1,1})_{s>0}$. Light tracing with next-event estimation simulates caustics more efficiently than path tracing because vertices resulting from specular scattering are directly projected on the image. However, light tracing also incurs high variance since virtual cameras generally have a small lens (eventually no lens for pinhole camera models), which is rare to reach by tracing random paths from light sources. Moreover, the closer path vertices are from the camera, the lower is their density after projection and many pixels are not filled with color, producing holes in the rendered image. A better alternative is to combine path tracing and light tracing with bidirectional path tracing, that takes advantage of all local path sampling strategies to build a low variance estimator.

4.3 BIDIRECTIONAL PATH TRACING

Bidirectional path tracing (BPT) was introduced by Lafortune et al. [LW93] and improved by Veach et al. [Vea97] with MIS. The basic idea of this method is to combine path tracing and light tracing in order to sample more efficiently the path space, taking advantage of all sampling strategies $(p_{s,t})_{s+t>1}$ defined in Section 4.1.3. Figure 20 demonstrates the efficiency of bidirectional path tracing on a complex configuration through a comparison with path tracing after the same rendering time.

4.3.1 *The bidirectional estimator*

A sequence of paths $\bar{\mathbf{x}}_{s,t}$, obtained with strategies $(p_{s,t})_{s+t=k}$, can be used to estimate the term $I_{i,k}$ of the measurement equation. MIS must be used to account for the fact that multiple sampling strategies are used to generate these paths, as illustrated by Figure 21. Moreover,



(a) Reference (b) Path tracing (c) Bidirectional path tracing

Figure 20 – Comparison between path tracing and bidirectional path tracing after 90 seconds.

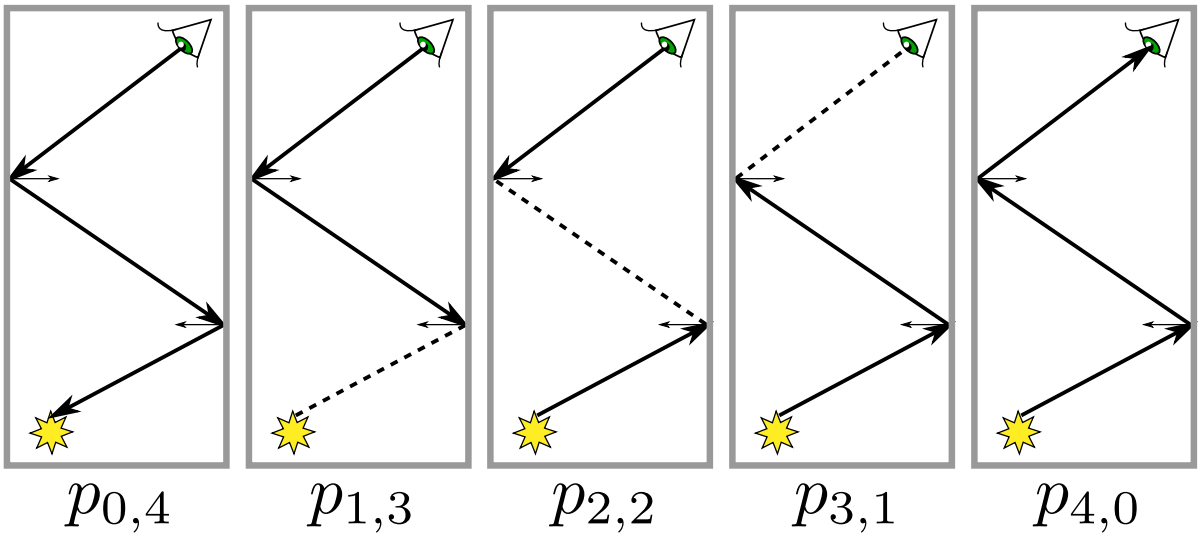


Figure 21 – Illustration of the 5 sampling strategies to obtain a path of length 4.

choosing a good heuristic for computing MIS weights, such as the balance heuristic, results in a low variance estimator. The bidirectional MIS estimator is:

$$\hat{I}_{i,k}^{\text{BPT}} = \sum_{s+t=k} w_{s,t}(\bar{\mathbf{x}}_{s,t}) \frac{f^{(i)}(\bar{\mathbf{x}}_{s,t})}{p_{s,t}(\bar{\mathbf{x}}_{s,t})} \tag{121}$$

The balance heuristic defines the weight $w_{s,t}(\bar{\mathbf{x}}_{s,t})$ by:

$$w_{s,t}(\bar{\mathbf{x}}_{s,t}) = \frac{p_{s,t}(\bar{\mathbf{x}}_{s,t})}{\sum_{s'+t'=k} p_{s',t'}(\bar{\mathbf{x}}_{s,t})} \tag{122}$$

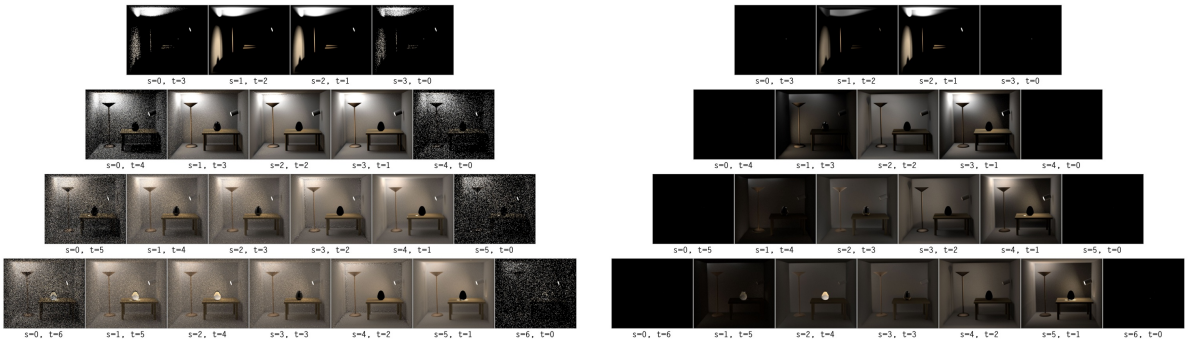
To efficiently evaluates these weights, Veach [Vea97, p. 305] introduced an iterative method that evaluate the weight of a path in $\mathcal{O}(k)$ time. More recently, Antwerpen [Ant11] proposed a recursive solution in $\mathcal{O}(1)$ time that only need information stored at the connection vertices $\mathbf{y}_s$ and $\mathbf{z}_t$.

4.3.2 *Limitations of BPT*

BPT is one of the most robust rendering algorithm thanks to the use of MIS with a large number of sampling strategies, as demonstrated by Figure 22. Nevertheless, BPT is subject to two main limitations: rendering of caustics caused by specular materials and efficiency in highly occluded environment.

4.3.2.1 *Caustics.*

The strength of BPT comes from its ability of sampling a path with many strategies that are blended with multiple importance sampling, to exploit the strength of each one. However, a path containing a specular event can only be sampled by a limited number of strategies since one edge is deterministic.



(a) Unweighted contributions (b) Weighted contributions

Figure 22 – Illustration of unweighted and weighted contributions of individual sampling strategies of BPT. The noise generated by each strategy is efficiently reduced using multiple importance sampling.

To solve this problem, Georgiev et al. [GKDS12] recently introduced the Vertex Connection and Merging algorithm, simultaneously with Hachisuka et al. [HPJ12] with their Unified Path Sampling method. These two solutions are equivalent in terms of implementation and combine BPT with progressive photon mapping (PPM) in a common sampling framework. PPM is a biased algorithm that is efficient for caustics, but not for diffuse or glossy scattering. Combining BPT with PM results in a robust rendering method, able to deal with all the most common surface scattering effects.

4.3.2.2 *Occlusions.*

The most expensive operation involved in BPT is the connection between a large number of pairs of surface points, which require many visibility tests. BPT does not importance sample the factors of the contribution function that results from the connection of two vertices. In particular, the visibility factor is likely to be null for many paths in highly occluded scenes, thus BPT is subject to high variance in this case.

Resampling (Section 3.3.4) is often used to address this problem, in order to choose randomly the terms of the estimator that have the best chance to give a high contribution.

Lafortune et al. [LW95] proposed to reduce the number of shadow rays traced to evaluate the bidirectional estimator (121). For that, they compute for each complete path $\bar{\mathbf{x}}_{s,t}$ a probability $P(\bar{\mathbf{x}}_{s,t})$ proportional to the unoccluded contribution of the path. Then a pair (s', t') is chosen based on these probability and only one shadow ray between $\bar{\mathbf{y}}_{s'}$ and $\bar{\mathbf{z}}_{t'}$ is traced. If the shadow ray is unoccluded, the sum of unoccluded contributions is used to estimate the pixel intensity. Otherwise, the pixel intensity is estimated by zero. This estimator is unbiased and can be used to improve the efficiency of BPT by tracing few shadow rays. However, ignoring visibility to compute the probabilities can introduce too much variance in highly occluded scenes since many pixel intensities would be estimated by zero in that case.

More recently, Popov et al. [PRDD15] introduced probabilistic connections for BPT. This method traces a high number of light sub-paths at the beginning of each iteration. Probability

mass functions over the light sub-paths are then built, based on their contribution to a sparse set of eye vertices, referred as *importance records*. When tracing an eye sub-path through a pixel, the importance record closest to each eye vertex is gathered and a single light path is resampled from the PMF computed for the importance record. This light path is used to estimate the complete sum of contributions. While this method has the advantage of taking into account visibility, parameters may be difficult to set (number of light paths and number of importance records) to achieve maximal efficiency.

In Chapter 11, we introduce a new method to resample connections in order to improve convergence rate of BPT, based on geometrical and topological information offered by a curvilinear skeleton of the empty space of the scene.

4.4 MANY-LIGHT RENDERING

Many-light methods approximate the radiance field with a finite distribution of points called *virtual point lights* (VPLs), obtained from the vertices of light paths. These VPLs are used to illuminate points seen by the camera. Therefore, global illumination is reduced to direct illumination, that can be efficiently performed on GPU.

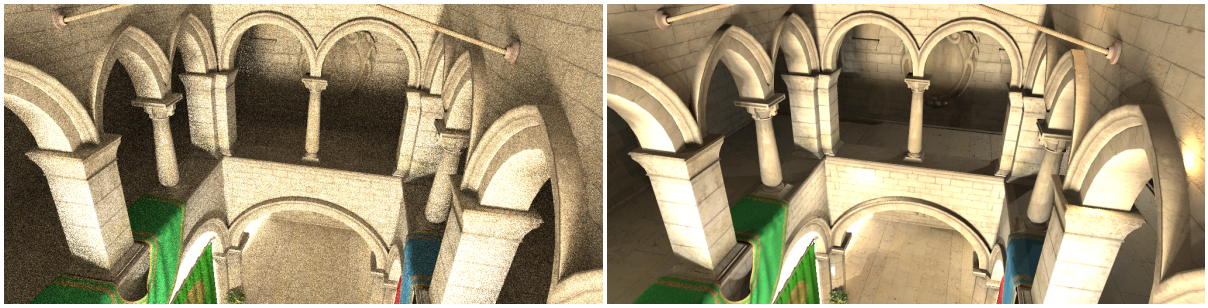
Many-light rendering was introduced by Keller [Kel97] with the *instant radiosity* algorithm. This method renders the image progressively by sampling a small number of VPLs at each iteration. Their contribution is accumulated using the GPU, with shadow mapping to test for visibility.

In this section we describe instant global illumination, which generalizes instant radiosity. For more details about recent works on many-light rendering, we refer the reader to the recent state-of-the-art report by Dachsbacher et al. [DKH+14].

4.4.1 *Instant global illumination*

Instant global illumination (IGI) samples the path space with strategies $(p_{s,2})_{s\geq 0}$. The second vertex of each eye path is connected to light paths of various length to approximate the measurement equation. Furthermore, the set of light paths is shared by all pixels and pixel measurements are highly correlated. This correlation induces a replacement of noise, usually present in Monte Carlo methods, by structural artifacts like hard shadows or bright spots of illumination when the number of light paths is not high enough (see Figure 23).

Let $(\bar{\mathbf{y}}_s^j)_{j\in[\![1,N]\!]}$ be N light paths sampled according to strategy $p_{L,s}$, $\bar{\mathbf{z}}_2$ an eye path sampled with strategy $p_{E,2}$ through pixel i and $(\bar{\mathbf{x}}_{s,2}^j)_{j\in[\![1,N]\!]}$ the sequence of complete paths obtained by connecting $\mathbf{z}_2$ with the last vertex of each light path.



(a) Bidirectional path tracing (b) Instant global illumination

Figure 23 – Comparison between bidirectional path tracing and instant global illumination after 90 seconds. While the error introduced by BPT estimation is expressed as noise, the error introduced by IGI appears as small plot of illuminations or hard shadows at some parts of the image.

The term $I_{i,2+s}$ from the measurement equation is estimated by:

$$\hat{I}^{\text{IGI}}_{i,2+s} = \frac{1}{N}\sum_{j=1}^{N}\frac{f^{(i)}(\bar{\mathbf{x}}^j_{s,2})}{p_{s,2}(\bar{\mathbf{x}}^j_{s,2})} \tag{123}$$

In the context of many-light rendering, light vertices are named virtual point lights (VPLs) since they acts as virtual light sources illuminating every eye vertex $\mathbf{y}_2$ directly visible from the camera. The estimator is often rewritten with the following equivalent expression:

$$\hat{I}^{\text{IGI}}_{i,2+s} = \sum_{j=1}^{N} I(\mathbf{y}^j_s)M(\mathbf{y}^j_s,\mathbf{z}_2)G(\mathbf{y}^j_s,\mathbf{z}_2)W(\mathbf{z}_2) \tag{124}$$

where the factors of the inner expression can be derived from the decomposition of the contribution function, detailed in Section 4.1.3:

— $I(\mathbf{y}^j_s)$ is the *intensity* of the VPL $\mathbf{y}^j_s$, expressed as:

$$I(\mathbf{y}^j_s) = \frac{f_L(\bar{\mathbf{y}}^j_s)}{N.p_L(\bar{\mathbf{y}}_s)} \tag{125}$$

— $M(\mathbf{y}^j_s,\mathbf{z}_2)$ is the *material factor*, accounting for BSDF factors at $\mathbf{z}_2$ and at $\mathbf{y}^j_s$ (or emitted radiance when the VPL is on a light source). Thus, this factor is such that:

$$C(\bar{\mathbf{x}}^j_{s,2}) = M(\mathbf{y}^j_s,\mathbf{z}_2)G(\mathbf{y}^j_s,\mathbf{z}_2) \tag{126}$$

— $W(\mathbf{z}_2)$ is the *sensor response intensity* of the eye vertex $\mathbf{z}_2$, expressed as:

$$W(\mathbf{z}_2) = \frac{f^{(i)}_E(\bar{\mathbf{z}}_2)}{p_E(\bar{\mathbf{z}}_2)} \tag{127}$$

The term $I_{i,2}$ is generally not estimated with virtual point lights but with rays leaving the camera that intersect a light source.

4.4.2 *Limitations of many-light rendering*

Many-light rendering is know to produce noticeable artifacts when the number of VPLs is not sufficiently high to represent the distribution of radiance all over the scene. These artifacts are the result of variance introduced by the strategies $p_{s,2}$ that are not efficient enough to represent robustly all kinds of lighting effects, such as short range illumination, glossy reflections and caustics. The fact that all pixel intensity measurements are highly correlated also involve that artifacts are not "hidden" by noise.

A major problem of strategies $p_{s,2}$ is the geometric factor between the measurement point and the VPL, which is not importance sampled by the strategy. This factor contains a division by the squared distance between the two points that can goes to zero at some specific geometric configurations such as the intersection of two perpendicular walls. Consequently, estimates are not bounded as well as their variance, producing small bright spots on rendered images. This problem is known as *weak singularity* and is often solved by clamping the geometric factor. This solution is simple but introduce some bias in the estimator, that does not cancel as more VPLs are sampled. Visually, this bias appears as images darker than their ground truth. Kollig et al. [KK04] propose to compensate the bias with path tracing when the geometric factor has to be clamped, in order to reintroduce the missing contribution in the estimate with a better sampling strategy. This approach effectively solves the bias problem but make the computation time unpredictable and introduce noise in images.

A related problem is glossy BSDFs, since material factors between the measurement point and the VPL are also not importance sampled by strategies $p_{s,2}$. Some methods attacked this problem by changing the virtual light source representation. Virtual spherical lights [HKWB09] (VSLs) have been proposed to replace VPLs. Such light sources allows to evaluate the solid angle they span at a given measurement point and to evaluate more efficiently the BSDF over a range of directions instead of a single one. Rich VPLs [SHD15] goes a step further, storing at each VPL a complete estimation of their incident radiance function on a texture. This texture allows to efficiently evaluate the complete glossy reflection at the VPL location and to get better estimates.

4.4.3 *Scalable many-light rendering*

To obtain a robust representation of the radiance field in complex scenes, a high number of VPLs need to be sampled (hundreds of thousands). In that case, the evaluation of their contribution becomes a real problem in terms of performances. Scalable evaluations solutions have been developed in order to reduce the number of VPLs needed to produce a realistic image.

Lightcut solutions [WFA+05, WABG06] build a tree of VPLs, effectively clustering them based on geometric and contribution properties. The evaluation of the radiance at an eye vertex corresponds to the computation of a cut in the tree that determine the VPLs to evaluate. This cut is obtained from thresholds given on the error produced by the approximation.

Matrix based solutions [HPB07, OP11] interprets the evaluation of VPLs contribution as a matrix where each column represents a VPL and each line a pixel. The cell (i, j) of the matrix is then the contribution of the j-th VPL to the i-th pixel's intensity. By studying the properties of this matrix and noticing that it usually has low rank, these methods are able to cluster VPLs and pixels such that few evaluations need to be computed in order to reconstruct a good approximation of the whole matrix.

For a given view configuration, all VPLs does not hold the same importance. In occluded scenes, most of them can even bring no contribution to the final image. Georgiev et al. [GS10] propose to discard VPLs using Russian roulette based on an approximation of their contribution to the image. This methods allows to keep the more meaningful VPLs only for the full evaluation and to produce a better image for the same amount of time. Importance caching [GKPS12] is a recent method based on resampling to reduce the number of VPL evaluations by computing their contribution at a few number of surface points visible from the camera, referred as importance records. From these contributions, a probability mass function is built at each importance record. The nearest importance records from a measurement point are gathered and their pmf are mixed in order to select few VPLs to perform the evaluation.

Bidirectional instant radiosity [SIMP06a] and Metropolis instant radiosity [SIP07] offer good sampling strategies to obtain the set of VPLs, according to their contribution to the final image.

4.4.4 *Real-time and interactive many-light rendering*

Rasterization pipelines implemented on graphical processing units (GPUs) are especially efficient to render direct illumination. Since many-light rendering transforms global illumination problem into direct illumination from multiple light sources, it can be efficiently implemented using specialized APIs such as OpenGL or Direct3D to reach real-time or interactive frame rates.

The generation of VPLs can be performed by rendering the scene in a texture from primary lights point of view, a technique called *reflective shadow maps* (RSM) [DS05]. This texture stores geometric and material information required to extract VPLs. This solution can only generate VPLs resulting from light paths on length 2, but it is generally considered enough to compute meaningful indirect illumination in real-time.

Once VPLs are obtained, the main bottleneck to compute their contribution is the computation of shadows. A robust and artifact-free rendering of global illumination requires a high number of VPLs, and the cost associated with shadows scales with the number of VPLs. Ignoring shadows can be a solution for indirect illumination as long as the scene does not present to much occlusions. When it is not the case, shadow maps related strategies are generally used. For thousands of VPLs, computing one full shadow map per VPL is not a option since each shadow map requires the rendering of the scene multiple times (up to 5 for cube shadow maps at surface VPLs).

To reduce the number of shadow maps that need to be computed, Dong et al. [DGR+09] compute clusters of VPLs using k-means and compute only one shadow map for each cluster, which is shared by VPLs of the cluster.

Another strategy is to compute an *imperfect shadow map* (ISM) [RGK+08, REH+11, BBH13] for each VPL. Such shadow map is a low resolution paraboloid shadow map computed by rasterizing an approximation of the scene obtained from a point cloud. This rasterization produces holes in the shadow map that are filled with a reconstruction process. This method produces artifacts that compensate each others as the number of VPL increases. However, the cost of computing ISMs remains dependent on the number of VPLs and the parameters involved in the method are not easy to set up for a given scene.

A related method is ManyLOD [HREB11], for *many level-of-details*, that renders paraboloid shadow maps for each VPL using a hierarchical representation of the point cloud representing the scene. This hierarchy is used to raster points of various size to the shadow map, depending on the distance from the VPL to the geometry. The drawbacks are similar to ISMs but the shadow maps generally have a better quality.

Recently, Olsson et al. [OSK+14] take advantage of new hardware capabilities to compute virtual shadow maps for each VPL. Their method is able to generate high quality shadows but only for hundreds of VPLs.

For real time rendering of glossy reflexion, Tokuyoshi proposed to replace the VPL representation with *virtual spherical gaussian lights* (VSGL) [Tok14]. A VSGL approximates the exitant radiance distribution of many VPLs using a gaussian mixture decomposition and allow better time coherency in the rendering of highly directional effects, such as glossy and specular scattering.

In Chapter 7 and Chapter 8, we present two methods we developed for real-time/interactive rendering of global illumination with VPLs, by approximating shadows casted by VPLs using a curvilinear skeleton of the empty space of the scene.

4.5 PHOTON MAPPING

Photon Mapping (PM), introduced by Jensen et al. [JC95, Jen96, Jen01], is a rendering algorithm that estimate reflected radiance using density estimation instead of Monte Carlo integration. PM is known to be particularly efficient for the rendering of caustics, but also to produce blurry estimates. In fact, the original method proposed by Jensen is biased and not consistent (it does not converge to the correct result). Progressive photon mapping is a reformulation of photon mapping that remain biased but offers a consistent estimator.

4.5.1 *Basic photon mapping*

To derive the PM estimator, we first rewrite the incident radiance at a point as the second-order derivative of the incident radiant power:

$$L_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}}) = \frac{d^2\Phi_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}})}{d\sigma^{\perp}_{\mathbf{x}}(\omega_{\mathrm{i}}) d\mathcal{A}(\mathbf{x})} \tag{128}$$

The expression of the reflected radiance can then be rewritten:

$$\begin{aligned} L_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{o}}) &= \int_{\omega_{\mathrm{i}} \in \mathcal{S}^2} L_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}}) f_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}}) d\sigma^{\perp}_{\mathbf{x}}(\omega_{\mathrm{i}}) \\ &= \int_{\omega_{\mathrm{i}} \in \mathcal{S}^2} \frac{d^2\Phi_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}})}{d\sigma^{\perp}_{\mathbf{x}}(\omega_{\mathrm{i}}) d\mathcal{A}(\mathbf{x})} f_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}}) d\sigma^{\perp}_{\mathbf{x}}(\omega_{\mathrm{i}}) \\ &= \int_{\omega_{\mathrm{i}} \in \mathcal{S}^2} \frac{d^2\Phi_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}})}{d\mathcal{A}(\mathbf{x})} f_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}}) \end{aligned} \tag{129}$$

This new expression can then be estimated using N random light vertices (referred as *photons*) located in a disk of radius R centered in $\mathbf{x}$:

$$L^{\mathrm{PM}}_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{o}}) = \frac{1}{\pi R^2} \sum_{i=1}^{N} f_{\mathrm{s}}(\mathbf{x}, \omega_i, \omega_{\mathrm{o}}) \Delta\Phi_i \tag{130}$$

Where $\Delta\Phi_i$ is the incident power carried by the i-th photon. In practice, photons are just non-specular vertices from a total of N_L light paths sampled before the rendering is actually performed, similarly to VPLs in many-light rendering. Given a light vertex $\mathbf{y}_s$ from a light path $\bar{\mathbf{y}}s$, its incident power is estimated by:

$$\Delta\Phi_{\mathrm{i}}(\mathbf{y}_s) = \frac{1}{N} \frac{f_L(\bar{\mathbf{y}}_s)}{p_L(\bar{\mathbf{y}}_s)} \tag{131}$$

Where f_L is the light sub-path partial contribution function and p_L the light sub-path sampling strategy (both defined in Section 4.1).

All photons are stored in a *photon map*, which is generally implemented as a Kd-tree or a hash grid, to allow fast range queries for the gathering of nearest photons during the rendering pass.

The PM estimate (130) is biased, and, as a consequence, gives a blurry estimation of the reflected radiance. However, we have the following convergence property:

$$L_{\mathrm{s}}(\mathbf{x}, \omega_{\mathrm{o}}) = \lim_{N_L \to \infty} \frac{1}{\pi R^2} \sum_{i=1}^{\lfloor N_L^{\alpha} \rfloor} f_{\mathrm{s}}(\mathbf{x}, \omega_i, \omega_{\mathrm{o}}) \Delta\Phi_i \tag{132}$$

In this equation, $\alpha \in (0, 1)$ is a constant number that ensures that the number of photons gathered around each point goes to infinity at a rate infinitely slower than N_L. This property is exploited by *progressive photon mapping*, an extension of PM which is also biased, but consistent.

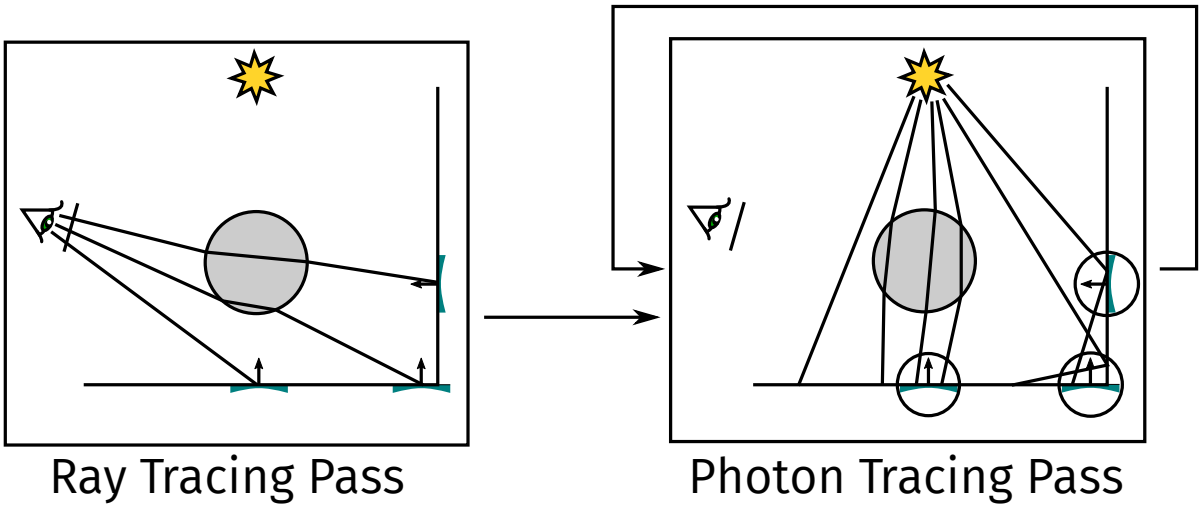


Figure 24 – Illustration of progressive photon mapping.

4.5.2 *Progressive photon mapping*

Progressive photon mapping [HOJ08] (PPM) is a reformulation of photon mapping that exploit the convergence property expressed by Equation (132). Even if storing an infinite number of photons in the photon map is not possible, we can still approximate the reflected radiance expression in a consistent way using a progressive algorithm. Instead of storing photons, PPM stores eye vertices (referred as *hit points*) obtained from eye paths that are stopped at the first non specular vertex. Each hit point H_i is associated to a disk of radius R_i that is used to gather photons. During photon tracing, each time a photon fall in the disk of a hit point H_i, its contribution is accumulated to the estimate associated to the hit point. Moreover, the radius R_i of the disk is reduced by a factor that depends on the total number of photons that have contributed to the reflected radiance of H_i, and the accumulated reflected radiance is rescaled to account for the radius reduction. After all passes, the contribution accumulated to each hit point is divided by the total number of photons and reported to the corresponding pixel. Figure 24 illustrates the computation of progressive photon mapping and Figure 25 shows a comparison between PPM, PM and several other rendering algorithms.

4.6 MARKOV CHAIN MONTE CARLO METHODS

Markov Chain Monte Carlo (MCMC) generates random samples for Monte Carlo integration based on a Markov chain. The first MCMC algorithm introduced in rendering is *Metropolis light transport* (MLT) [VG97].

(a) PT (b) BPT (c) MLT (d) PM (e) PPM

Figure 25 – Comparison of photon mapping and progressive photon mapping against several rendering solutions. While PM offers a blurry estimation, PPM provides a consistent strategy that converges to the real image.

4.6.1 *Metropolis sampling*

The idea behind *Metropolis sampling* [MRR+53] is to simulate a Markov chain whose distribution is proportional to a function $f : \Omega \to \mathbb{R}^+$. More specifically, we want to generate a sequence of samples $X_0, ..., X_n$ such that X_{i+1} only depends on X_i and $\lim_{i\to\infty} p_i = p$, where p_i is the PDF of X_i and $p = f / \int_\Omega f$ is the PDF proportional to f. To sample X_{i+1} from X_i, we use a conditional PDF K such that $K(y|x)$ is the probability density of having $X_{i+1} = y$ knowing that $X_i = x$. Each PDF p_i satisfies:

$$p_i(y) = \int_{x\in\Omega} K(y|x)p_{i-1}(x)dx \tag{133}$$

K is referred as the *transition function*. The goal is to construct K such that $\lim_{i\to\infty} p_i = p$. To achieve that goal, we use a *tentative transition function* T, where $T(y|x)$ gives the probability density of having $X_{i+1} = y$ given that $X_i = x$. Given a proposal $Y \sim T(\cdot, X_i)$, we let:

$$X_{i+1} = \begin{cases} Y \text{ with probability } a(Y|X_i) \\ X_i \text{ with probability } 1 - a(Y|X_i) \end{cases} \tag{134}$$

where a is an *acceptance probability function*. To ensure that $p_i \to p$ as $i \to \infty$, a must respect a condition named *detailed balance*:

$$f(x)T(y|x)a(y|x) = f(y)T(x|y)a(x|y) \tag{135}$$

To reach equilibrium as fast as possible [VG97], the acceptance probability must be defined by:

$$a(y|x) = \min\left\{1, \frac{f(y)T(x|y)}{f(x)T(y|x)}\right\} \tag{136}$$

4.6.2 *Metropolis Light Transport*

Veach et al. [VG97] applied Metropolis sampling to light transport simulation, resulting in the *Metropolis light transport* (MLT) algorithm. For this, the contribution function must be rewritten:

$$f_i(\bar{\mathbf{x}}) = h_i(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \tag{137}$$

where h_i is the pixel filter function at pixel i and f is the *image contribution function*, which is not null only for paths that contribute to at least one pixel. A Monte Carlo estimator for the measurement I_i is then:

$$\bar{I}_i^{\text{MLT}} = \frac{1}{N} \sum_{j=1}^{N} \frac{h_i(\bar{\mathbf{x}}_j) f(\bar{\mathbf{x}}_j)}{p(\bar{\mathbf{x}}_j)} \tag{138}$$

If we suppose that $p = f/I$ with $I = \int_{\mathcal{P}} f(\bar{\mathbf{x}}) d\bar{\mathbf{x}}$ (the total image brightness), we have:

$$\bar{I}_i^{\text{MLT}} = \frac{I}{N} \sum_{j=1}^{N} h_i(\bar{\mathbf{x}}_j) \tag{139}$$

Sampling according to p is a difficult problem, but Metropolis sampling can be used to generate a Markov chain whose distribution converges to p. A tentative transition function on paths is defined by Veach et al. [VG97] as a mixture of several mutation strategies. Each strategy is tailored for a particular type of lighting effect.

Since the distribution of samples from the Markov chain only approaches p as their number grows to infinity, a form of bias named *startup bias* is introduced in the estimation process. To eliminate it, contributions must be weighted according to the initial path $\bar{\mathbf{x}}_0$ of the Markov chain. This path is obtained by sampling a strategy p_0 (for example bidirectional path sampling) and its contribution $h_i(\bar{\mathbf{x}}_0)$ is weighted by the factor $W_0 = f(\bar{\mathbf{x}}_0)/p_0(\bar{\mathbf{x}}_0)$ (note that this weight is an unbiased estimate of I). The weight for subsequent samples $\bar{\mathbf{x}}_j$ of the chain is simply set to $W_j = W_0$ The complete MLT estimator is:

$$\bar{I}_i^{\text{MLT}} = \frac{1}{N} \sum_{j=1}^{N} W_j h_i(\bar{\mathbf{x}}_j) = \frac{W_0}{N} \sum_{j=1}^{N} h_i(\bar{\mathbf{x}}_j) \tag{140}$$

A proof that this estimator is unbiased is given by Veach et al. [VG97]. However, the unbiasedness of the estimator does not guarantee its consistency. Indeed, given an initial sample $\bar{\mathbf{x}}_0$, $\bar{I}_i$ does not converge to I_i since W_0 is not equal to I except if $p_0 = p$, which is not the case (otherwise Metropolis sampling is not required). In order to make the estimator consistent, we can average the estimates computed for M independent Markov chains. This new estimator is defined by:

$$\begin{aligned} \bar{I}_i^{\text{MLT}} &= \frac{1}{M} \sum_{j=1}^{M} \bar{I}_i^{(j)} \\ \bar{I}_i^{(j)} &= \frac{W_0^{(j)}}{N} \sum_{k=1}^{N} h_i(\bar{\mathbf{x}}_k^{(j)}) \end{aligned} \tag{141}$$

As $M \to \infty$ and $N \to \infty$, this estimator converges to I_i. Another possibility is to resample the estimator with one path $\bar{\mathbf{x}}_0^{(J)}$ chosen among the sequence of initial paths according to the discrete probabilities $P_j = W_0^{(j)} / \sum_{k=1}^{M} W_0^{(k)}$. The resampled estimator is:

$$\begin{aligned} \bar{I}_i^{\text{MLT}} &= \frac{1}{P_J.M} \bar{I}_i^{(J)} \\ &= \frac{\sum_{k=1}^{M} W_0^{(k)}}{W_0^{(J)}.M} \frac{W_0^{(J)}}{N} \sum_{k=1}^{N} h_i(\bar{\mathbf{x}}_k^{(J)}) \\ &= \frac{1}{M} \sum_{k=1}^{M} W_0^{(k)} \frac{1}{N} \sum_{k=1}^{N} h_i(\bar{\mathbf{x}}_k^{(J)}) \end{aligned} \tag{142}$$

In this equation, the first factor $\frac{1}{M} \sum_{k=1}^{M} W_0^{(k)}$ is an unbiased estimate of I with M samples obtained according to p_0. Note that this last version of MLT can be seen as a two-stage algorithm: initially, M paths are sampled according to p_0 in order to estimate I. Then one is randomly chosen based on their image contribution to start the Markov chain and to complete the estimation.

Despite the fact that MLT samples paths distributed according to the path contribution function (after enough iterations of the Markov chain), it is known to be difficult to implement due to the mutation strategies on path space. To ease the implementation of MLT, Kelemen et al. [KSK01] proposed to apply path mutations on the primary sample space, which is the unit hypercube of random numbers sampled to generate random paths.

Also, while MLT efficiently renders bright parts of the image, it generally neglects dark parts since the exploration of the path space is focused on contributing paths. Obtaining an error uniformly distributed over the image is therefore difficult with MLT.

Part II

SKELETONS AND OPENING MAPS FOR GLOBAL ILLUMINATION

5

OVERVIEW OF SKELETONIZATION

Most of our contributions use a *topological curvilinear skeleton of the empty space of the scene*, represented by a 3D graph embedded in empty space.

The concept of skeleton was introduced by Blum [Blu67] through an intuitive analogy to fire propagation. Suppose an object made up of grass; if we set fire to the border of the object, the skeleton would be constituted of the meeting points of the flame fronts. This definition corresponds in the continuous framework $\mathbb{R}^n$ to the *medial axis* of the object, defined as the set of points which are centers of maximal balls (balls included in the object and not strictly included in any other such ball). An important topological property of the medial axis in $\mathbb{R}^n$ is that it is *homotopic* to the original object [Lie03]. More generally, the medial axis in continuous framework satisfies the four requirements, expressed by Hilditch [Hil69], that a skeleton should met for the purpose of shape analysis:

— It should be homotopic to the original object.
— It should have a lower dimension than the original object.
— It should be centered in the original object.
— It should be stable: skeletonizing a skeleton should give this same skeleton.

The first condition expresses that it should be possible to deform continuously the original object to its skeleton, that is, without cutting or merging any part of the object. This property ensures, in any dimension, that the number of connected components of the object is conserved. In 3D, it also guarantee that cavities and holes of the object are still present in the skeleton. This requirement is important to study topological properties of the object from its skeleton.

In the digital framework $\mathbb{Z}^n$, the medial axis does not always satisfy the first and second properties. To overcome this problem, various methods have been proposed to perform skeletonization of digital objects.

This chapter offers a brief overview of skeletonization methods based on *thinning*, the process of removing points from a discrete objects in order to obtain a skeleton. The result is a subset of the original object that should satisfy the four mentioned requirements.

Contents

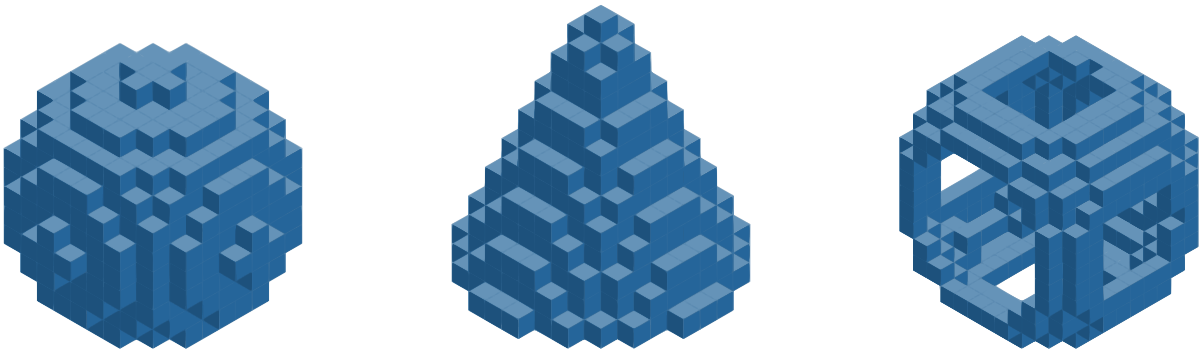


Figure 26 – Illustration of digital objects in 3D. Each voxel is represented by a cube.

5.1 THINNING IN THE DIGITAL FRAMEWORK

5.1.1 *Digital objects*

Digital objects are the basic entities studied by *digital geometry* and *digital topology* [AK04], two important fields of discrete shape analysis. A n-dimensional digital object X is a subset of $\mathbb{Z}^n$, the digital grid. A digital object X is *bounded* if $|X| \in \mathbb{N}$. The *complementary* of X is denoted $\bar{X} := \mathbb{Z}^n \setminus X$. In 3D, points of $\mathbb{Z}^3$ are called *voxels*. Figure 26 illustrates digital objects of $\mathbb{Z}^3$.

A point $\mathbf{x} \in \mathbb{Z}^n$ is defined by its coordinates $(x_1, ..., x_n)$ with $x_i \in \mathbb{Z}$, $\forall i \in [\![1, n]\!]$.

5.1.2 *Neighborhood*

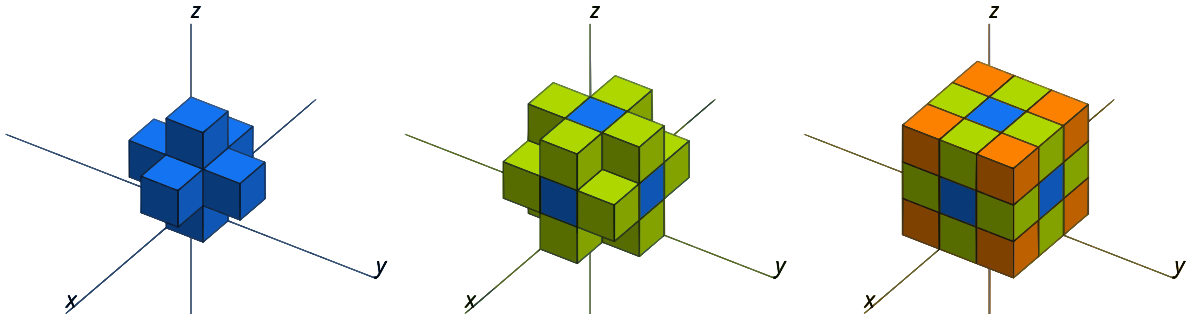
The topology of a digital object is strongly related to an *adjacency relation*, defined between points of $\mathbb{Z}^n$. Such relation is equivalent to the definition of a k-neighborhood $\mathcal{N}_k(\mathbf{x}) \subset \mathbb{Z}^n$ for each point $\mathbf{x} \in \mathbb{Z}^n$.

In 2D, neighborhoods generally considered for the digital grid are:
- the 4 neighborhood of $\mathbf{x}$ is the set $\mathcal{N}_4(\mathbf{x}) = \{\mathbf{y} \in \mathbb{Z}^2 \mid d_{\text{euc}}(\mathbf{x}, \mathbf{y}) \leq 1\}$
- the 8 neighborhood of $\mathbf{x}$ is the set $\mathcal{N}_8(\mathbf{x}) = \{\mathbf{y} \in \mathbb{Z}^2 \mid d_{\text{euc}}(\mathbf{x}, \mathbf{y}) \leq \sqrt{2}\}$

In 3D, neighborhoods generally considered for the digital grid are:
- the 6-neighborhood of $\mathbf{x} \in \mathbb{Z}^3$ is the set $\mathcal{N}_6(\mathbf{x}) = \{\mathbf{y} \in \mathbb{Z}^3 \mid d_{\text{euc}}(\mathbf{x}, \mathbf{y}) \leq 1\}$
- the 18-neighborhood of $\mathbf{x} \in \mathbb{Z}^3$ is the set $\mathcal{N}_{18}(\mathbf{x}) = \{\mathbf{y} \in \mathbb{Z}^3 \mid d_{\text{euc}}(\mathbf{x}, \mathbf{y}) \leq \sqrt{2}\}$
- the 26-neighborhood of $\mathbf{x} \in \mathbb{Z}^3$ is the set $\mathcal{N}_{26}(\mathbf{x}) = \{\mathbf{y} \in \mathbb{Z}^3 \mid d_{\text{euc}}(\mathbf{x}, \mathbf{y}) \leq \sqrt{3}\}$

where $d_{\text{euc}}(\mathbf{x}, \mathbf{y})$ refers to the *euclidean distance* between two points. Figure 27 illustrates the 3D neighborhoods.

For a k-neighborhood, we define $\mathcal{N}_k^*(\mathbf{x}) := \mathcal{N}_k(\mathbf{x}) \setminus \{\mathbf{x}\}$. For $X \subset \mathbb{Z}^k$, we define $\mathcal{N}_{X,k}(\mathbf{x}) := \mathcal{N}_k(\mathbf{x}) \cap X$ and $\mathcal{N}_{X,k}^*(\mathbf{x}) := \mathcal{N}_k^*(\mathbf{x}) \cap X$.



(a) 6-neighborhood (b) 18-neighborhood (c) 26-neighborhood

Figure 27 – Illustration of standard 3D neighborhoods in the digital framework.

5.1.3 *Connectivity*

The notion of k-connected components of a digital object can be defined from the notion of neighborhood.

Definition 5.1. Let $X \subset \mathbb{Z}^n$ and $\mathbf{x}, \mathbf{y} \in X$. A *k-path* from $\mathbf{x}$ to $\mathbf{y}$ in X is a sequence of voxels $(\mathbf{p}_1, ..., \mathbf{p}_n)$ such that:

— $\mathbf{p}_1 = \mathbf{x}$ and $\mathbf{p}_n = \mathbf{y}$
— $\forall i \in [\![1, n]\!]$, $\mathbf{p}_i \in X$
— $\forall i \in [\![1, n-1]\!]$, $\mathbf{p}_{i+1} \in \mathcal{N}_k(\mathbf{p}_i)$

When such a sequence exists, we note $\mathbf{x} \overset{X,k}{\longleftrightarrow} \mathbf{y}$.

Definition 5.2. We define $C_{k,X}(\mathbf{x})$, the *k-connected component* of X containing $\mathbf{x}$, as the maximal subset of X such that:

— $\mathbf{x} \in C_{k,X}(\mathbf{x})$
— $\forall \mathbf{y} \in C_{k,X}(\mathbf{x})$, $\mathbf{x} \overset{X,k}{\longleftrightarrow} \mathbf{y}$

Figure 28 illustrates connected components of a two dimensional digital object.

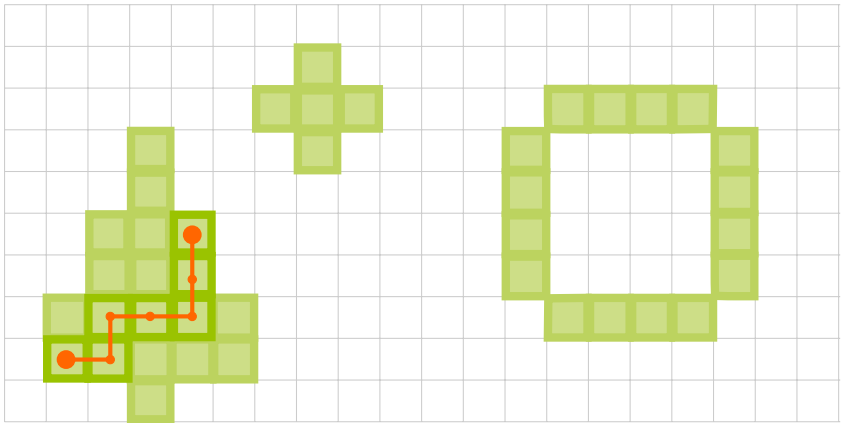


Figure 28 – A digital object of $\mathbb{Z}^2$. This object has six connected components for the 4-adjacency, but only three for the 8-adjacency. Indeed, the rightmost shape is not 4-connected. A 4-path is illustrated between two points of the leftmost shape.

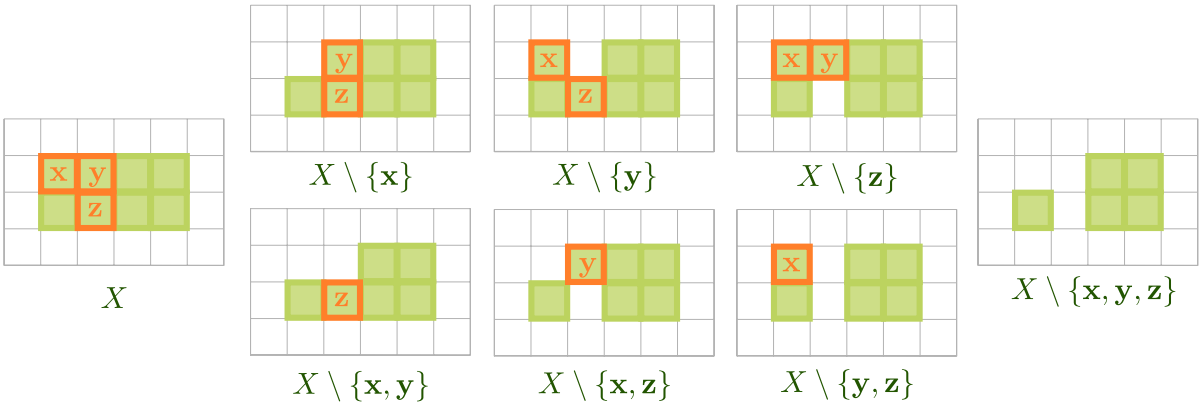


Figure 29 – The points $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in X$ are all 4-simple and 8-simple since removing them (top images) does not change the topology of the object for these two adjacencies. The pair $\{\mathbf{x}, \mathbf{y}\}$ can be removed in parallel while still maintaining the 4 and 8-topology. However, removing the pair $\{\mathbf{x}, \mathbf{z}\}$ does not conserve the 4-topology of the object since it produces a new connected component for the 4-adjacency. Deletion of the pair $\{\mathbf{y}, \mathbf{z}\}$ does not conserve either the 4 or 8-topology.

5.1.4 *Thinning algorithms*

In the digital framework, thinning algorithms remove *simple points* from the original object in order to compute a skeleton. Intuitively, a point is *k-simple* if it can be removed from an object without changing its topology for the k-adjacency relation. In 3D, this is equivalent to the conservation of the number of connected components, cavities and tunnels. 2D simple points are illustrated in Figure 29.

Various characterizations of simple points have been proposed, and it has been shown that 2D, 3D and 4D k-simple points can be locally characterized in constant time [Kon97, CB08, BC09, CB09].

We give here a local criterion to identify 6-simple and 26-simple points in 3D, by the mean of *topological numbers* [BM94].

Definition 5.3. Let $X \subset \mathbb{Z}^3$, $\mathbf{x} \in X$ and $k \in \{6, 26\}$. The *geodesic neighborhood* $G_k(\mathbf{x}, X)$ is defined as follow:

— $G_6(\mathbf{x}, X) := \mathcal{N}^*_{X,26}(\mathbf{x}) \bigcap \bigcup_{\mathbf{y} \in \mathcal{N}^*_{X,6}(\mathbf{x})} \mathcal{N}_6(\mathbf{y})$

— $G_{26}(\mathbf{x}, X) := \mathcal{N}^*_{X,26}(\mathbf{x})$

Definition 5.4. Let $X \subset \mathbb{Z}^3$, $\mathbf{x} \in X$ and $k \in \{6, 26\}$. The *topological number* $T_k(\mathbf{x}, X)$ is defined as the number of connected components of $G_k(\mathbf{x}, X)$.

Proposition 5.5. *Let $X \subset \mathbb{Z}^3$ and $\mathbf{x} \in X$.*

— *The point $\mathbf{x}$ is 26-simple for X iff $T_{26}(\mathbf{x}, X) = 1$ and $T_6(\mathbf{x}, \bar{X}) = 1$.*

— *The point $\mathbf{x}$ is 6-simple for X iff $T_6(\mathbf{x}, X) = 1$ and $T_{26}(\mathbf{x}, \bar{X}) = 1$.*

With the possibility of identifying efficiently simple points of an object, a thinning algorithm removes them until no more can be found. In order to preserve interesting visual features of the original object, a *constraint set* can be given to the algorithm. All points contained in this

set are constrained to be kept in the resulting skeleton. This set can also be updated during the algorithm, as new features of the object get detected.

Two main strategies exist for removing simple points: sequential strategies and parallel strategies.

5.1.4.1 *Sequential removal*

A sequential thinning algorithm removes simple points one at a time, until no more can be found. Obtaining a centered skeleton with a sequential algorithm can be difficult since, in order to do so, points must be removed from the object one layer at a time, starting at the border. Figure 30 demonstrates the difference between a centered skeleton and a skeleton obtained by removing random simple points at each step.

A widely used strategy to obtain a centered skeleton with a sequential thinning algorithm is to compute a priority function on the object and to remove simple points according to this function, taking the point with lowest priority at each step. The euclidean distance map can be used as priority function, by removing at each step the simple point with the lowest possible value. Since this map gives high values to points located at the center of the object, the resulting skeleton is centered.

5.1.4.2 *Parallel removal*

Parallel thinning algorithms remove multiple simple points at the same time. This strategy allows to obtain a centered skeleton without relying on a priority function since points are naturally removed "layer by layer". Another advantage of parallel thinning algorithms is the possibility of implementing the algorithm on a parallel processor such as a GPU, in order to achieve maximal efficiency.

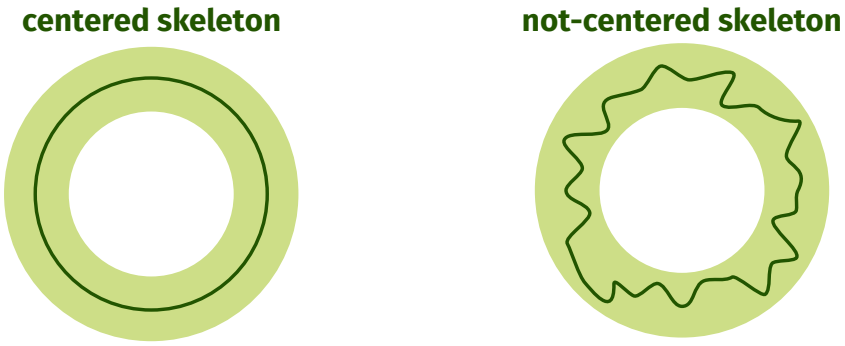


Figure 30 – Obtaining a centered skeleton (left image) with a sequential thinning algorithm requires to use a priority function. Otherwise, the order of removal is random and the skeleton may not keep the visual shape of the original object (right image), while still being homotopic to the object.

However, removing simple points simultaneously can change the topology of the original object, as illustrated in Figure 29. Indeed, when a simple point is removed, three situations can occur:

— Simple points may become non-simple.
— Non-simple points may become simple.
— Nothing changes.

Identifying in 3D a large set of simple voxels that can be removed simultaneously is a difficult problem and several theories have been developed to check if a parallel thinning algorithm effectively preserve topology and to help designing new correct algorithms. Among them, the *critical kernel framework* transposes results from the cubical complex framework to the digital framework and provide new characterizations of simple points as well as efficient parallel thinning algorithms of digital objects.

5.2 THINNING IN THE CUBICAL COMPLEX FRAMEWORK

Analyzing discrete objects in the digital framework actually involves several problems related to geometry and topology [Cha10, page. 122].

The notion of dimensionality of a set of voxel is hard to define properly since a voxel is, by definition, a volume. Consequently, the only way of defining a digital surface is to identify local configurations that traduce more or less the digitalization of a continuous surface [CB14]. With such kind of definition, important properties of the continuous framework may not be conserved and the analysis of a shape through its digital representation can lead to incoherences. For example, the intersection of two digital surfaces might not always define a digital curve. More generally, defining and identifying *digital manifold* in $\mathbb{Z}^n$, with the same properties as their continuous equivalent, is an extremely hard problem that leads to the intuition that the digital framework is not well suited to analyze geometric features of discrete objects.

Another problem, related to topology, is the fact that the adjacency relation chosen for a digital object must not be the same than the adjacency relation used on the complementary of this object [KMW91, ML00, Loh01]. Indeed, using the same adjacency for both sets invalidates the fundamental *Jordan theorem*, stating that a closed simple curve always separates the continuous plane $\mathbb{R}^2$ in two connected components, an interior and an exterior. Figure 31 illustrates that this is not always the case for a digital curve when the adjacency relation is the same for the curve and its complementary. This paradox constraints the use of a valid adjacency pair $(k, \bar{k})$ - where the k-adjacency is used for the object and the $\bar{k}$-adjacency is used for complementary - in order to define topological features when it is important to consider an object and its interior/exterior components. In 2D, the valid pairs are $(4, 8)$ and $(8, 4)$. In 3D, they are $(6, 26)$, $(26, 6)$, $(6, 18)$ and $(18, 6)$. A possible way to avoid this paradox is to change the partitioning of the plane in order to introduce adjacency relations that maintain the Jordan theorem without the use of a pair. The hexagonal grid is an example, but does not generalize well to higher dimension and is less practical than the square grid in terms of implementation.

The *cubical complex framework* constitutes an interesting alternative since it represents discrete objects by finite sets of continuous elements of various dimensions, with a structure defining

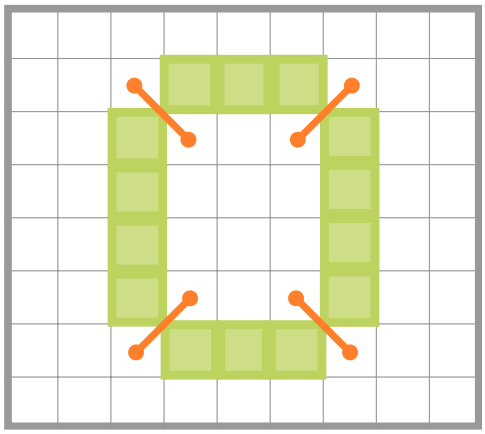


Figure 31 – A two dimensional digital object, represented by green squares. If we consider the 8-adjacency for this object, then it forms a closed simple curve that should separate the complementary of the object (white squares) in two connected components. However, the orange lines demonstrates that the complementary is still 8-connected. If we consider the 4-adjacency for this object, then it is constituted of four connected components and is not a closed simple curve anymore. But in this case, the complementary becomes disconnected in 4-adjacency. The only correct way of having the Jordan theorem (and its reciprocal) in this grid is to choose a different adjacency relation for the object and its complementary.

a unique adjacency relation that matches the continuous case. While thinning can directly be applied to cubical complexes [CC09, Cha10, Cou11], the framework is also useful to derive results and algorithms on the digital topology framework [Ber07, CB09, BC09, BC14, BC14].

5.2.1 *Basic definitions*

In the voxel framework, discrete objects are made of voxels. In the cubical complex framework, discrete objects are made of cubes, squares, lines and vertices. Such representation allows to characterize precisely the dimension of an object: a surface does not contain any cube, and a curve does not contain any cube or square.

To define properly these notions, we consider the family of sets $\mathbb{F}_0^1$ and $\mathbb{F}_1^1$ defined by:

$$\mathbb{F}_0^1 := \{\{a\} \mid a \in \mathbb{Z}\} \tag{143}$$

$$\mathbb{F}_1^1 := \{\{a, a+1\} \mid a \in \mathbb{Z}\} \tag{144}$$

Definition 5.6. (face) Any subset $f \subset \mathbb{Z}^n$ such that f is the Cartesian product of m elements of $\mathbb{F}_1^1$ and $(n-m)$ elements of $\mathbb{F}_0^1$ is called a *face* or, more precisely, a *m-face* of $\mathbb{Z}^n$, with m the dimension of f. We write $dim(f) = m$.

A 0-face is called a *vertex*, a 1-face is an *edge*, a 2-face is a *square*, and a 3-face is a *cube*. Given $m \in [\![0, n]\!]$, we denote by $\mathbb{F}_m^n$ the set composed of all m-faces in $\mathbb{Z}^n$. We denote by $\mathbb{F}^n$ the set composed of all faces in $\mathbb{Z}^n$:

$$\mathbb{F}^n := \bigcup_{m=0}^{n} \mathbb{F}_m^n \tag{145}$$

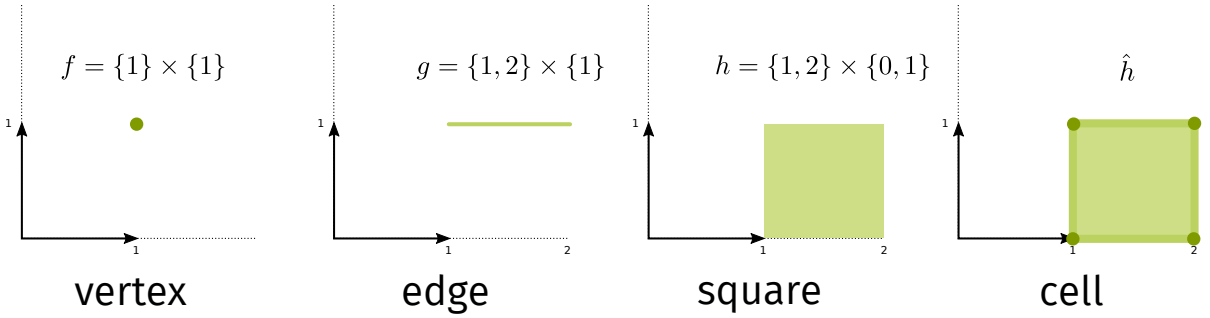


Figure 32 – Illustrations of some faces and a cell of $\mathbb{Z}^2$.

Contrary to the digital framework, the adjacency relation between faces is defined uniquely. The neighborhood $\mathcal{N}(f)$ of a face $f \in \mathbb{F}^n$ is:

$$\mathcal{N}(f) := \{g \in \mathbb{F}^n \mid f \cap g \neq \emptyset\} \tag{146}$$

Definition 5.7. Let $f \in \mathbb{F}^n$. We set:

$$\begin{aligned} \hat{f} &:= \{g \in \mathbb{F}^n \mid g \subseteq f\} \\ \hat{f}^* &:= \hat{f} \setminus \{f\} \end{aligned} \tag{147}$$

Any element of $\hat{f}$ (resp. $\hat{f}^*$) is *a face of* f (resp. *a proper face of* f). A set of faces $X \subset \mathbb{F}^n$ is a *cell*, or *m-cell*, if there exists $f \in \mathbb{F}^n$ such that $X = \hat{f}$.

Figure 32 illustrates faces and a cell of $\mathbb{Z}^2$.

Example 5.1. *If* $f = \{0,1\} \times \{2\} = \{(0,2),(1,2)\}$ *(*f *is an edge of* $\mathbb{Z}^2$*), then* $\hat{f} = \{\{0\} \times \{2\}, \{1\} \times \{2\}, \{0,1\} \times \{2\}\} = \{\{(0,2)\}, \{(1,2)\}, \{(0,2),(1,2)\}\}$

The *closure* of a set of faces X is the set:

$$\widehat{X} := \cup\{\hat{f} \mid f \in X\} \tag{148}$$

Definition 5.8. A finite set X of faces in $\mathbb{F}^n$ is a *cubical complex* if $X = \widehat{X}$, and we write $X \preceq \mathbb{F}^n$.

Intuitively, a set of faces X is a complex if for each face $f \in X$, X also contains all the sub-faces contained in f, i.e. the faces $g \in \hat{f}$.

A face $f \in X$ is *a facet of* X if f is not a proper face of any face of X. The *dimension of* X is $\dim(X) = \max\{\dim(f) \mid f \in X\}$. If $\dim(X) = d$, then we say that X is a d-complex.

5.2.2 *From digital objects to cubical complexes*

To transpose a digital object $X \subseteq \mathbb{Z}^n$ to the cubical complex framework, we associate to each point $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{Z}^n$ an n-face $\Psi(\mathbf{x}) \in \mathbb{F}^n$ defined by:

$$\Psi(\mathbf{x}) = \{x_1, x_1 + 1\} \times \ldots \times \{x_n, x_n + 1\} \tag{149}$$

We extend the map Ψ to sets to obtain a set of n-faces associated to X: $\Psi(X) = \{\Psi(\mathbf{x}) \mid \mathbf{x} \in X\}$. The cubical complex associated to X is then defined as the closure $\widehat{\Psi(X)}$ of $\Psi(X)$.

5.2.3 *The collapse operation*

The *collapse operation* is the basic operation for performing homotopic thinning of a complex, and consists of removing free pairs of faces:

Definition 5.9. Let $X \preceq \mathbb{F}^n$, and let f, g be two faces of X. The face g is *free for* X, and the pair (f, g) is *a free pair for* X if f is the only face of X which strictly contains g.

It can be easily seen that if (f, g) is a free pair for a complex X, then f is a facet of X and $\dim(g) = \dim(f) - 1$.

Definition 5.10. Let $X \preceq \mathbb{F}^n$, and let (f, g) be a free pair for X. The complex $X \setminus \{f, g\}$ is an *elementary collapse of* X.

Let $Y \preceq \mathbb{F}^n$. The complex X *collapses onto* Y if there exists a sequence of complexes $(X_0, ..., X_\ell)$ of $\mathbb{F}^n$ such that $X = X_0$, $Y = X_\ell$ and for all $i \in \{1, \dots, \ell\}$, X_i is an elementary collapse of X_{i-1}. We also say, in this case, that Y *is a collapse of* X.

5.2.4 *Parallel directional thinning*

In the cubical complex framework, parallel removal of free pairs can be easily achieved by following simple rules that we give now. First, we need to define the *direction* and the *orientation* of a free face. Let (f, g) be a free pair for $X \preceq \mathbb{F}^n$: we have $\dim(g) = \dim(f) - 1$, and it can be seen that $g = f \cap f'$, where f' is the translate of f by one of the $2n$ vectors of $\mathbb{Z}^n$ which have all their coordinates equal to 0 except one, which is either equal to +1 or -1. Let v be this vector, and c its non-null coordinate. We define $Dir(f, g)$, called the *direction* of the free pair (f, g), as the index of c in v. The *orientation* of the free pair (f, g) is defined as $Orient(f, g) = 1$ if $c = 1$, and $Orient(f, g) = 0$ else.

The following proposition, proven by Chaussard and Couprie [CC09], gives a necessary and sufficient condition for removing two free pairs of faces in parallel from a complex, while preserving topology.

Proposition 5.11. *Let $X \preceq \mathbb{F}^n$, and let (f, g) and (k, ℓ) be two distinct free pairs for X. The complex X collapses onto $X \setminus \{f, g, k, \ell\}$ if and only if $f \neq k$.*

So we can remove two free pairs from X in parallel without changing the topology as long as their facets are different.

From Proposition 5.11, the following corollary is immediate.

Corollary 5.12. *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X such that, for all $a, b \in \{1, \dots, m\}$ (with $a \neq b$), $f_a \neq f_b$. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

Considering two distinct free pairs (f, g) and (i, j) for $X \preceq \mathbb{F}^n$ such that $Dir(f, g) = Dir(i, j)$ and $Orient(f, g) = Orient(i, j)$, we have $f \neq i$. From this observation and Corollary 5.12, we deduce the following property.

Algorithm 2 : [CC09]. *ParDirCollapse*(X, W, ℓ)

Data : A cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed, and $\ell \in \mathbb{N}$, the number of layers of free faces which should be removed from X

Result : A cubical complex

```
1  while there exists free faces in X \ W and ℓ > 0 do
2      L = Border(X);
3      for t = 1 → n do
4          for s = 0 → 1 do
5              for d = n → 1 do
6                  E = {(f, g) free for X | g ∉ W, Dir(f, g) = t, Orient(f, g) = s, dim(f) = d};
7                  G = {(f, g) ∈ E | f ∈ L and g ∈ L};
8                  X = X \ G;
9      l = l − 1;
10 return X;
```

Corollary 5.13. *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X having all the same direction and the same orientation. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

Intuitively, we want our thinning algorithm to remove free faces of a complex "layer by layer" to avoid unequal thinning of the input complex. Therefore, we want each execution of the algorithm to remove free faces located on the border of the input complex. We define $Border(X)$ as the set all faces belonging to a free pair for X. We now introduce Alg. 2, a directional parallel thinning algorithm.

On a single execution of the main loop of Alg. 2, only faces located on the border of the complex are removed (l. 7). Thanks to corollary 5.13, we can remove faces with same direction and orientation in parallel (l. 8), while guaranteeing topology preservation. Figure 33 depicts the first steps of the algorithm.

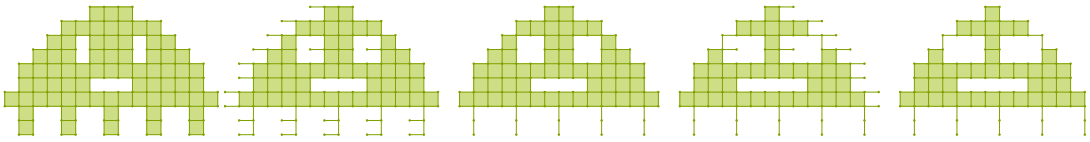


Figure 33 – Four first iterations of Alg. 2 running on the left-most shape.

Different definitions of orientation and direction can be given, each corresponding to a different order of free faces removal in the complex and leading to different results. Algorithm 2 can be implemented to run in linear time complexity (proportionally to the number of faces in the complex). Indeed, checking if a face is free or not may be easily done in constant time and when a free pair (f, g) is removed from the input complex, it is sufficient to scan the faces contained in f in order to find new free faces.

5.3 SKELETONS

5.3.1 *Ultimate skeletons*

An ultimate skeleton is obtained when no constraint set is defined while applying the thinning procedure. At the end the the process, the skeleton is minimal and cannot be thinned anymore since it contains no more simple point (in the digital framework) or no more simple pair (in the cubical complex framework). An ultimate skeleton has the advantage of being minimal while still encoding the topology of the original object. However, the global appearance of the object might be lost. For example, an elongated corridor with an S shape is reduced to a point (see Figure 34).

5.3.2 *Aspect preservation, surface and curvilinear skeletons*

The aspect preservation of the object is generally mandatory for most applications. Otherwise, the analysis of geometric features of the original object from its skeleton might be not possible. In our case, we want to extract a skeleton of the empty space of the scene. Regarding its application to rendering and visibility problems, it is essential to preserve the aspect of the empty space in order to get meaningful information at each point of the scene.

To obtain a skeleton that mimics the visual shape of a digital object, a *constraint set* has to be specified in order to keep some simple points safe from deletion. Generally, two strategies are possible to achieve this goal: find, during the skeletonization process, points whose neighborhood configuration seems interesting and keep them in the result [CCZ07, HR08, CCT10a], or choose, before skeletonization, interesting points of the object which should remain untouched, based on a function on these points and a filtering parameter [BC06, Pal07]. A possible constraint set is the digital medial axis, since this object is well centered in the object and had a similar visual appearance. A problem with this approach is that

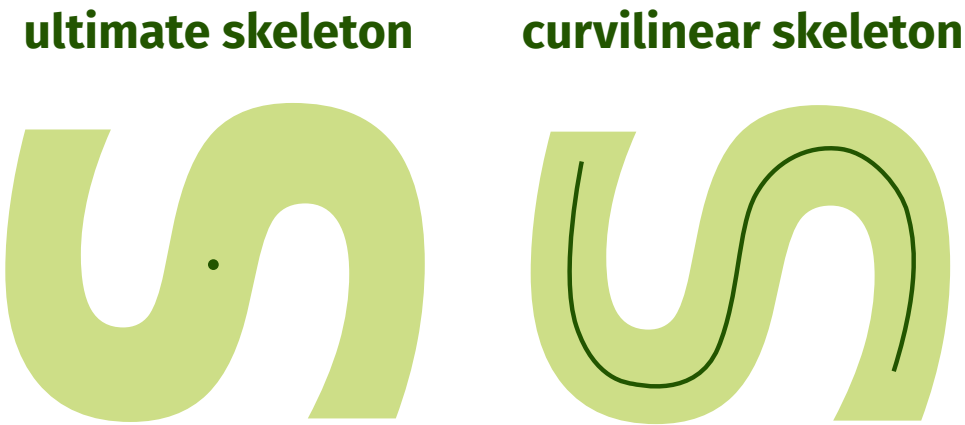


Figure 34 – An ultimate skeleton does not provide much information about the shape of the original object (left image). For geometric analysis purpose, a curvilinear skeleton is better (right image).

the medial axis is known to be very sensitive to noise, and thus generally contains many "spurious branches" that need to be filtered [Loh01, Chapter. 11]. The *discrete λ-medial axis* is a filtered medial axis that was proved to remain stable under small perturbations [CCT09] and which can be used as constraint set instead of the medial axis [CCT10b].

For aspect preservation of cubical complexes, Chaussard proposed in its PhD thesis [Cha10] a parameter-free method to obtain a robust skeleton that mimics the shape of the original object. This strategy evaluates a measure of the lifespan of each face and compares it to a measure of its centering in order to decide if the face represents an important feature of the original object. We present in Chapter 6 a variation of this method that we use to compute our skeleton of the empty space of the scene.

It is generally interesting to keep an information about the dimensionality of the object at different parts of its skeleton. Some volumetric regions of a 3D object may be more elongated in some directions than others, a feature that can be represented by a surface skeleton. Symmetric regions can be completely reduced to a curve, elongated enough to traduce the length of the object. In the digital framework, characterizing surfaces and curves is difficult, as discussed in Section 5.2. In 1998, Palágyi and Kuba [PK98] proposed a directional thinning algorithm to extract a curvilinear skeleton by performing 6 sub-iterations in each direction of the digital grid. The concept of "isthmuses", derived from the cubical complex framework, allows to detect parts of an object that are locally like a curve or a surface [RC11, CB14, BC15].

In the cubical complex framework, Chaussard and Couprie [CC09] proposed a solution to identify surfaces created during the execution of the basic thinning algorithm 2. Faces from these surfaces are put in the constraint set dynamically and kept safe from future deletion.

6

THINNING THE EMPTY SPACE IN THE CUBICAL COMPLEX FRAMEWORK

Most of the rendering algorithms proposed in this thesis are based on a curvilinear skeleton of the empty space of the 3D scene. This chapter presents the algorithm we developed to compute this skeleton, as well as details concerning the way we associate any surface point of the scene to a point of the skeleton. Section 6.1 gives some motivations regarding the use of a skeleton for light transport simulation. The thinning algorithm is detailed in Section 6.2, that computes a curvilinear skeleton in the cubical complex framework. This algorithm was presented at the DGCI 2013 conference [CNBC13], together with an application to path tracing detailed in Chapter 9. We propose in Section 6.3 a simple solution to filter the skeleton in order to reduce the number of its nodes, according to the geometry of the empty space. This filtering is required for some of our contributions for which the time complexity and memory requirement depends strongly on the number of nodes of the skeleton. Finally, we explain how we map any surface point to a skeleton's node in Section 6.4.

Contents

6.1 SKELETONS FOR LIGHT TRANSPORT SIMULATION

Skeletons are mainly applied on fields that rely on shape analysis, such as text recognition. By computing the skeleton of each letter of a text, it becomes easier to recognize it based on topological and geometrical properties. Another application of skeletons is segmentation and labeling, in order to partition an object in regions based on certain criteria, such as curvilinear or surface parts. Skeletonization is also widely used in medical imaging applications, as detailed by the recent survey published by Saha et al. [SSB15]

The contributions presented in this thesis employ a skeleton for an unusual purpose: light transport simulation. This section explains the motivation behind our works and how we extract a discrete representation of the empty space of the scene.

6.1.1 *Motivation*

Few methods explicitly use geometric or topological information about the scene to drive rendering algorithms. This can be explained by the fact that, most often, 3D scenes are represented by soup of triangles from which it is hard to extract any useful information. For light transport simulation, we are interested in the *empty space of the scene*, since light propagates in this object. As a volumetric object, implicitly defined by its boundaries (the surfaces of the scene), the empty space is dense and hard to describe precisely. A skeleton provides a sparse representation of the empty space as well as topological and geometric features describing its shape. The topology of the skeleton indicates what holes light paths can travel in the empty space, as well as connected components that are useful to identify light sources not accessible from the camera. In terms of geometry, the skeleton tells us which parts of the empty space are curved. Such regions generally require more reflections for a path to connect the camera with a light source. By storing information at skeleton points, such as the radius of the maximal ball centered in each node and inscribed in empty space, we can quickly extract an information about the thickness of the empty space at a specific point of the scene. This information is useful since narrow regions are usually difficult to explore with local path sampling solutions. The skeleton can also be used as a lightweight data structure to store information about the lighting distribution in the scene, or about the visibility between points. Our contributions aim at providing practical solutions to use this skeleton in the context of rendering.

To the best of our knowledge, our works constitute the first application of skeletonization to rendering problems.

6.1.2 *Digital empty space*

We are interested in thinning a *digital empty space* $E_{\mathrm{D}} \subset \mathbb{Z}^3$, since light propagates in the empty space of the scene. To compute this object, the 3D scene $\mathcal{M}$ is first voxelized [SS10] to obtain the *digital scene* $\mathcal{M}_{\mathrm{D}}$. This voxelization process is controlled by the choice of a

resolution (w, h, d), that defines the size of an axis-aligned grid $G_{w,h,d} := [\![1, w]\!] \times [\![1, h]\!] \times [\![1, d]\!] \subset \mathbb{Z}^3$ containing the digital scene.

The digital empty space is then defined as:

$$E_{\mathrm{D}} := G_{w,h,d} \setminus \mathcal{M}_{\mathrm{D}} \tag{150}$$

Note that we do not define the digital empty space as the complementary of the digital scene because we want to obtain a bounded object.

6.1.3 *What kind of skeleton and thinning algorithm ?*

A explained in Chapter 5, different kinds of skeleton can be computed depending on the information we want to extract from the original object. In our case, the empty space is a volume and we we basically have a choice between a pure curvilinear skeleton or a surface skeleton. For the works presented in this thesis, we decided to limit ourselves to a *curvilinear skeleton of the empty space* for its simplicity. While losing substantial geometric information about the empty space, a curvilinear skeleton has the advantage of being sparser and being efficiently representable with a graph (an array of nodes and edges). We acknowledge however, that a surface skeleton could provide more information about the empty space's geometry, and thus lead to more robust rendering methods. Consequently, we let the application of surface skeletons to rendering for future works.

Regarding the thinning algorithm, we decided to work in the cubical complex framework to obtain the skeleton. As previously said, this framework is well suited to represent discrete objects without loosing information about their dimensionality. This is required to guarantee that the skeleton produced by our method is curvilinear, whenever it is possible (some objects are not topologically equivalent to a curve, we discuss this potential issue in Section 6.2.4). Another advantage of the cubical complex framework is the existence of parameter-free thinning algorithms [Cha10], such as the one presented in this chapter. 3D scenes used in rendering can be quite complex, and having the thinning process depends on parameters could eventually make our algorithms not usable in practice without fine of tuning of parameters, which could be discouraging. Consequently, the only parameter controlling the production of our skeleton is the voxelization resolution, which can be easily determined based on the level of details we want to include in the digital representation of the empty space.

In order to apply the thinning algorithm presented in this section to the digital empty space E_{D}, we convert it using the function presented in Section 5.2.2, obtaining the *cubical complex empty space* $\Psi(E_{\mathrm{D}})$.

6.2 ASPECT PRESERVATION DURING THINNING

For rendering, the skeleton of the empty space needs to capture the main geometric features of the original scene. For example, if the scene is a corridor, the skeleton should be a line following the main direction of the corridor and centered in its empty space.

The simple thinning scheme (Algorithm 2) presented in Section 5.2.4 does not necessarily preserve geometrical features of the input object in the resulting skeleton (for example, the skeleton of a corridor could be reduced to a single vertex). In the following, we introduce a new method in the cubical complex framework, requiring no user input, for obtaining a curvilinear skeleton yielding satisfactory geometrical properties. Our method finds, during thinning, elements with a specific neighborhood configuration, and uses a function on these elements to decide whether to preserve them, or not, in the result. This thinning algorithm is based on the parameter-free method proposed by Chaussard in its PhD thesis [Cha10], but with improvements regarding the production of a curvilinear skeleton in 3D presenting few spurious branches.

6.2.1 *The lifespan of a face*

In the following, we define additional functions in the cubical complex, related to the thinning process (Section 5.2.4, Algorithm 2), which are essential for the definition of the constraint set to provide to the thinning algorithm. The first one we present is the *death date* of a face.

Definition 6.1. Let $f \in X \preceq \mathbb{F}^n$. The *death date of f in X*, denoted by $Death_X(f)$, is the smallest integer δ such that $f \notin ParDirCollapse(X, \emptyset, \delta)$.

Intuitively, the death date of a face indicates how many layers of free faces should be removed from a complex X, using Algorithm 2, before removing completely the face from X. We now define the *birth date* of a face:

Definition 6.2. Let $f \in X \preceq \mathbb{F}^n$. The *birth date of f in X*, denoted by $Birth_X(f)$, is the smallest integer b such that either f is a facet of $ParDirCollapse(X, \emptyset, b)$, or $f \notin ParDirCollapse(X, \emptyset, b)$.

The birth date indicates how many layers of free faces must be removed from X with Algorithm 2 before transforming f into a facet of X (we consider that a face "lives" when it is a facet). Finally, we define the *lifespan* of a face :

Definition 6.3. Let $f \in X \preceq \mathbb{F}^n$. The lifespan of f in X is the integer

$$Lifespan_X(f) := \begin{cases} +\infty & \text{if } Death_X(f) = +\infty \\ Death_X(f) - Birth_X(f) & \text{otherwise} \end{cases}$$

These three values depend on the order of direction and orientation chosen for Algorithm 2.

The lifespan of a face f of X indicates how many iterations this face "survives" as a facet in X, when removing free pairs with Algorithm 2, and is a good indicator of how important a face can be in an object. Typically, the higher the lifespan is, and the more representative of an object's geometrical feature the face is. The lifespan, sometimes called *saliency*, was used in [LCLJ10] (with the name "medial persistence") in order to propose a thinning algorithm in cubical complexes based on two parameters.

6.2.2 *Distance map, opening function and decenterness map*

In addition to the lifespan of a face, the proposed homotopic thinning method uses information on distance between faces in order to decide if a face should be kept safe from deletion. We define hereafter various notions based on distances in the voxel framework.

We set $d_1(x, y)$ as the L1 distance between x and y (Manhattan distance). Let $S \subset \mathbb{Z}^n$, for all $x \in \mathbb{Z}^n$, the map $D_1(S) : \mathbb{Z}^n \to \mathbb{N}$ is such that $D_1(S)(x) = \min_{y \in \bar{S}} d_1(x, y)$.

The *maximal* 1*-ball in* S *centered on* x is the set $\mathbb{MB}^1_S(x) = \{y \in \mathbb{Z}^n \mid d_1(x, y) < D_1(S)(x)\}$. We set, for all $x \in S$, the map $\Omega_1(S) : \mathbb{Z}^n \to \mathbb{N}$ such that $\Omega_1(S)(x) = \max_{x \in \mathbb{MB}^1_S(y)} D_1(S)(y)$: this value indicates the radius of a largest maximal 1-ball contained in S and containing x. If $x \in \bar{S}$, we set $\Omega_1(S)(x) = 0$. The map $\Omega_1(S)$ is known as the opening function of S based on the 1-distance (also called the granulometry function) [Mat67]: it allows to compute efficiently results of morphological openings by balls of various radius, and gives information on the local thickness of an object.

Given $S \subset \mathbb{Z}^n$, the value of $\Omega_1(S)(x)$ of every $x \in S$ can be naively computed by performing successive morphological dilations of values of the map $D_1(S)$. A linear algorithm for computing the map $\Omega_1(S)$ (with regard to the size of the input image) was proposed in [Cha10].

Finally, we define the *decenterness map*:

Definition 6.4. Given $S \subset \mathbb{Z}^n$, the *decenterness map of* S is the map $\mathcal{DC}_1(S) = \Omega_1(S) - D_1(S)$.

An example of these maps is shown on Figure 35.

In order to extend all these previous maps defined in $\mathbb{Z}^n$ to the cubical complex framework, we use the map Ψ^{-1}, inverse of the bijective map $\Psi : \mathbb{Z}^n \to \mathbb{F}^n_n$ defined in Section 5.2.2. It is used to project any n-face of $\mathbb{F}^n$ into $\mathbb{Z}^n$. This map induces a map from $\mathcal{P}(\mathbb{F}^n_n)$ to $\mathcal{P}(\mathbb{Z}^n)$, that we also denote by Ψ^{-1}.

Given $Y \subset \mathbb{F}^n$, we set $S = \Psi^{-1}(Y \cap \mathbb{F}^n_n)$. We define the map $D^{cc}_1(Y) : \mathbb{F}^n \to \mathbb{N}$ as follows: for all $f \in \mathbb{F}^n$,

$$D^{cc}_1(Y)(f) = \begin{cases} D_1(S)(\Psi^{-1}(f)) & \text{if } f \text{ is an } n\text{-face} \\ \max_{f \in \hat{g}^* \cap \mathbb{F}^n_n} D^{cc}_1(Y)(g) & \text{otherwise} \end{cases}$$

Informally, if f is a 3-face, then $D^{cc}_1(Y)(f)$ is the length of the shortest 1-path between the voxel "corresponding" to f and the set of voxels corresponding to $\bar{Y}$. In the same way, we define $\Omega^{cc}_1(Y)$ and $\mathcal{DC}^{cc}_1(Y)$.

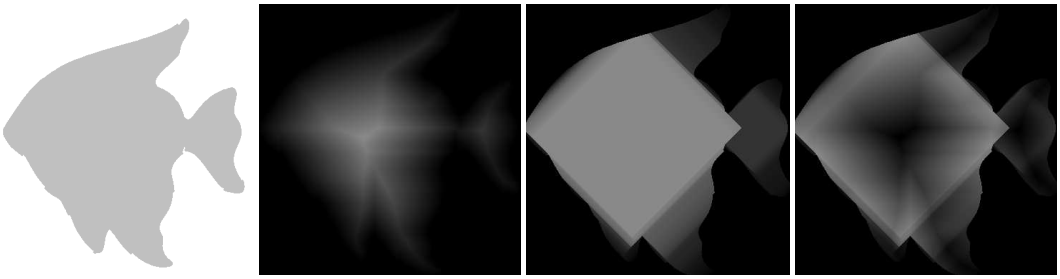


Figure 35 – **Examples of opening and decenterness map -** From left to right: a shape $S \subset \mathbb{Z}^2$ (in gray), $D_1(S)$, $\Omega_1(S)$ and $\mathcal{DC}_1(S)$ (low values have dark colour).

6.2.3 *Parameter-free filtered thinning*

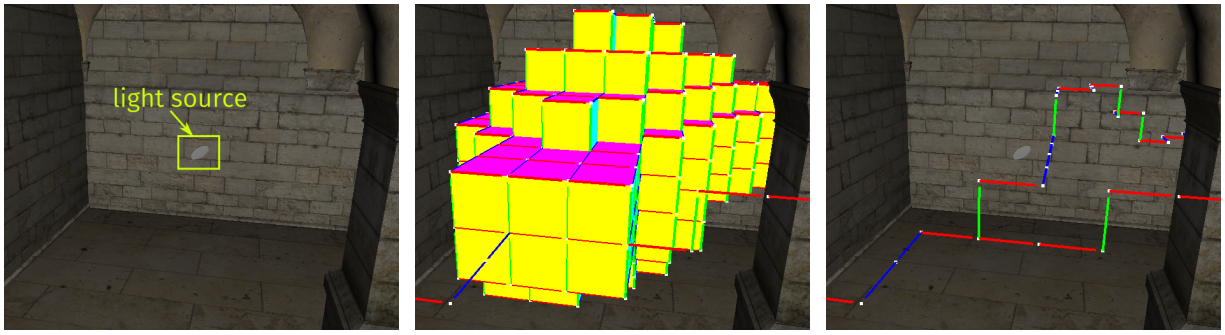
As previously said, we add edges to the constraint set W of Algorithm 2 in order to retain, in the resulting curvilinear skeleton, important edges from the original object. Given a cubical complex X, if an edge of X has a high decenterness value for X, then it is probably located too close to the border of X and does not represent an interesting geometrical feature to preserve. On the other hand, if an edge has a high lifespan for X, then it means it was not removed quickly, after becoming a facet, by the thinning algorithm and might represent some precious geometrical information on the original object. An idea would be to keep, during thinning, all edges whose lifespan is superior to the decenterness value. Unfortunately, this strategy produces skeletons with many spurious branches in surfacic areas of the original object.

We can identify surfacic areas of a complex as zones where squares have a high lifespan. Therefore, in order to avoid spurious branches in surfacic areas, we need to make it harder for edges to be preserved in these zones. It can be achieved by deciding that an edge will be kept safe from deletion by the thinning algorithm if its lifespan is superior to the decenterness value plus the lifespan of squares "around" this edge. This leads us to proposing Algorithm 3.

In order to understand what was realised on line 1 of Algorithm 3, we might point out that the birth date of an edge corresponds to the highest death date of the squares containing this edge. Moreover, the map $D_1^{cc}(X)$ gives, for all 3-faces of X, their death date (as the thinning algorithm naturally follows this map to eliminate cubes from a 3-complex). Therefore, for an edge f of X, $D_1^{cc}(X)(f)$ informs us on the highest death date of cubes containing f, also equal to the highest birth date of squares containing f. In conclusion, $Birth_X(f) - D_1^{cc}(X)(f)$ is an approximation of the lifespan of the squares containing f.

Algorithm 3 : $CurvilinearSkeleton(X)$

Data : A cubical complex $X \preceq \mathbb{F}^3$
Result : A cubical complex $Y \preceq \mathbb{F}^3$
1 $W = \{f \in X | Lifespan_X(f) > \mathcal{DC}_1^{cc}(X)(f) + Birth_X(f) - D_1^{cc}(X)(f) \text{ and } \dim(f) = 1\}$;
2 **return** $ParDirCollapse(X, W, +\infty)$;



(a) a light source floating (b) surface skeleton (c) curvilinear skeleton after piercing

Figure 36 – Illustration of a surface skeleton resulting from a cavity in the empty space (b), produced by a light source floating in the air (a). Piercing the surface part and reapplying the thinning algorithm on the pierced skeleton allows to obtain a curvilinear skeleton (c).

6.2.4 *Handling non-curvilinear skeletons*

Although the output of Algorithm 3 may contain 2-faces, the algorithm is said to be a *curvilinear skeletonization* algorithm because it only adds 1-faces (edges) in the constraint set W. However, for our rendering applications, we require a curvilinear skeleton even when the original object is not homotopic to a curve. This situation occurs even in simple situations, for example by putting a triangle floating in the empty space in order to define a light source. In that case, a cavity is created is the empty space, producing a surface enclosing the triangle in the skeleton (Figure 36b).

The only way of getting rid of surface parts of the skeleton is to make a compromise and to "break" the topology of the skeleton. However, instead of just removing 2-faces from the cubical complex skeleton before converting it to a graph, we pierce each connected surface part (by removing only one 2-face), and we reapply the thinning procedure. This choice allows to keep a connection between curves connected to given a surface part and thus to keep encoded in the curvilinear skeleton the information of accessibility between different regions of empty space (Figure 36c).

6.2.5 *Results*

Algorithm 3 allows to obtain a curvilinear skeleton from a three dimensional complex. The results presented in Figure 37 show that the skeletons contain the main geometrical information from the input shapes, and no spurious branches.

THINNING OF THE EMPTY SPACE The result of the thinning algorithm applied to the empty space of various scenes are demonstrated in Figures 38, 39, 40 and 41. For these scenes, the skeleton is directly shown as its graph representation embedded in the 3D scene.

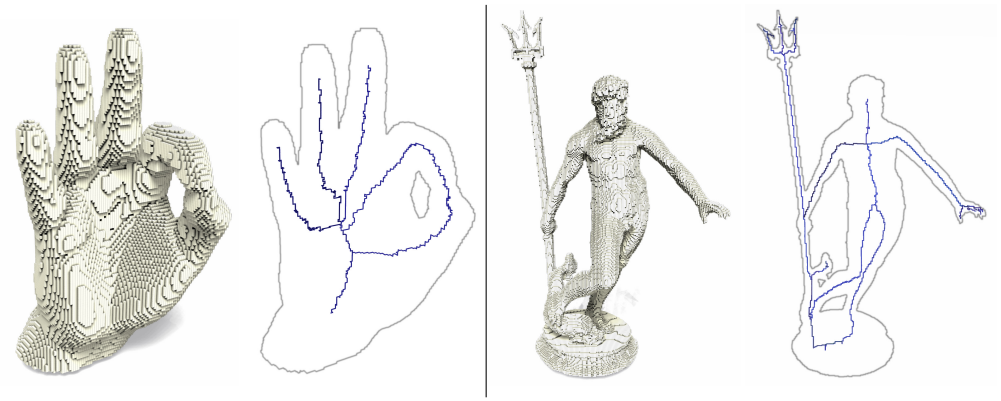


Figure 37 – **Results of algorithm 3** for two shapes: a hand (left) and a statue (right). In each pair, the rightmost image represents the skeleton.

SKELETON COMPUTATION TIME. The skeleton computation time is an important factor for its viability regarding applications to rendering. Fortunately, the method has a linear $\mathcal{O}(n)$ time complexity, with n being the number of faces of the cubical complex [Cha10]. Table 1 records time required to compute the skeleton of the empty space of various scenes, for several voxelization resolution. These measures, computed an Intel Core i7-3770K CPU 3.50GHz (8 cores), include both the voxelization time and thinning time.

While these times does not allow real-time thinning (and thus prevent dynamic geometry for real-time rendering algorithms based on the skeleton), they are negligible for off-line rendering algorithms, that are usually meant to run for hours in order to compute visually pleasing images (noise-free for path traced based algorithms). Also note that our implementation is not completely optimized and parallelized. A GPU version of the algorithm could potentially allows real-time thinning to be possible for reasonable voxelization resolutions. However, the basic implementation is not very cache-friendly, thus implementing the algorithm on the GPU would require a reformulation and some adaptation to really take advantage of the massively parallel computational power of the GPU.

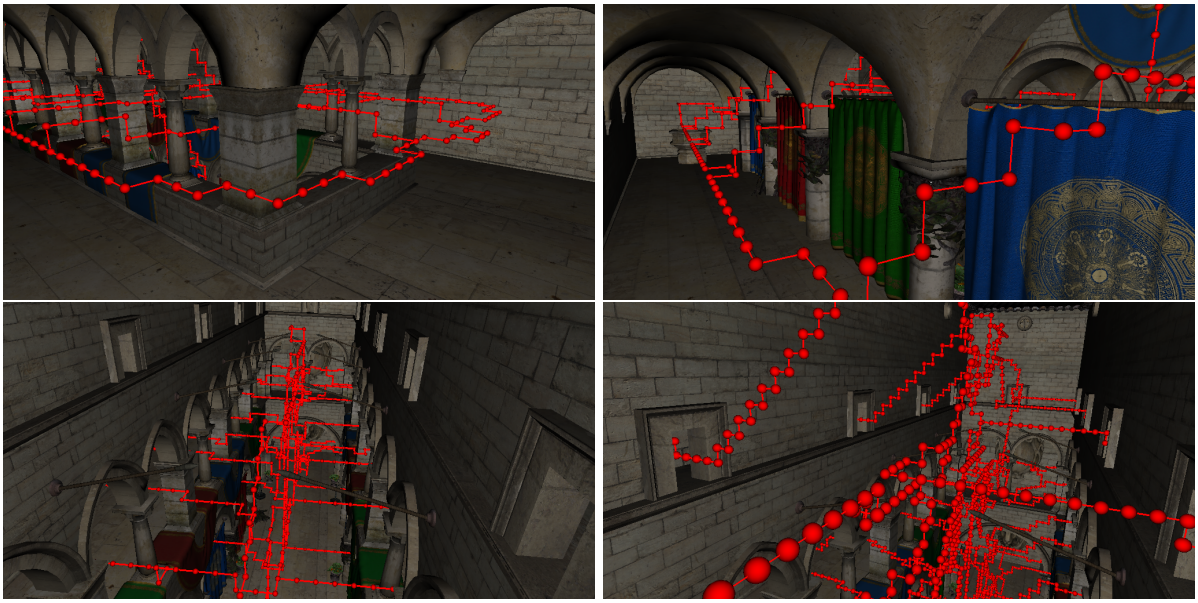
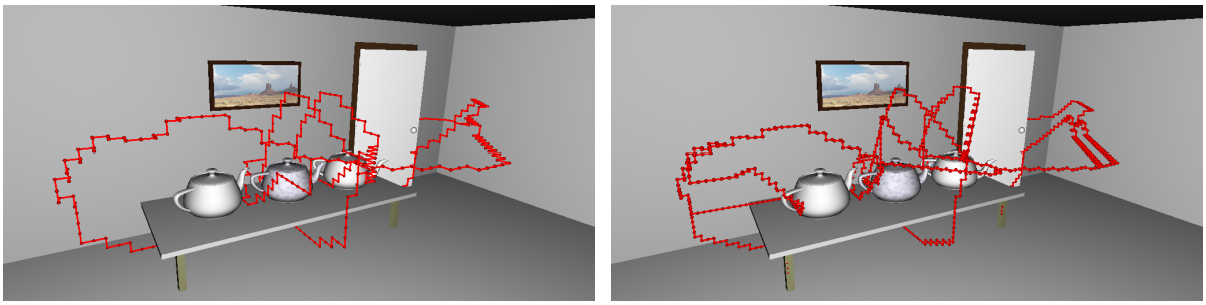
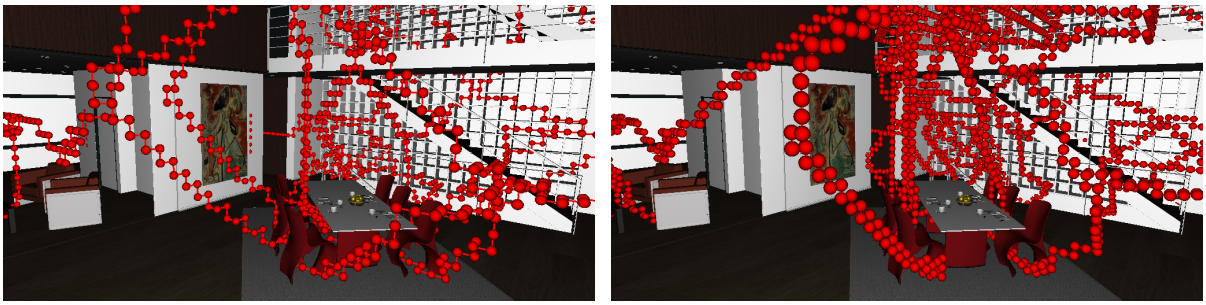


Figure 38 – Result of the thinning algorithm on the sponza scene, for a (119, 51, 73) resolution for the digital empty space. In order to cover the small apertures on top of the scene, the resolution of the voxelization grid must be increased to (234, 98, 144) (bottom-right image).



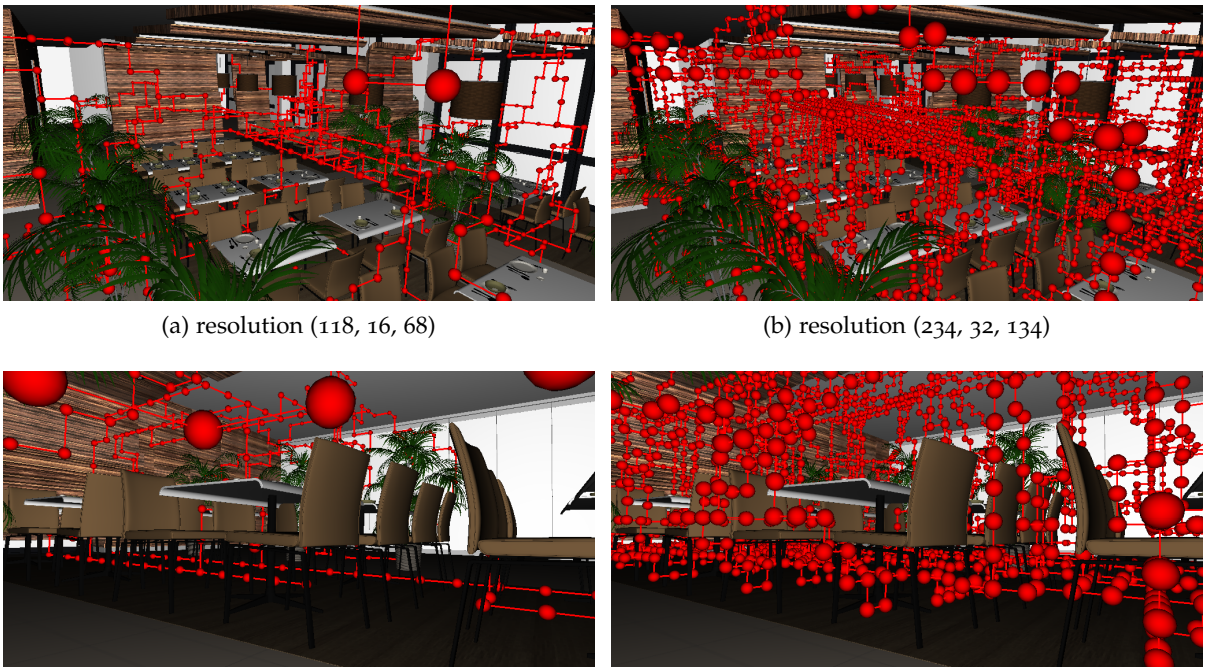
(a) resolution (72, 118, 28) (b) resolution (144, 234, 54)

Figure 39 – Skeleton of the veach door scene.



(a) resolution (234, 128, 152) (b) resolution (364, 200, 238)

Figure 40 – Skeleton of the apartment scene.



(a) resolution (118, 16, 68)

(b) resolution (234, 32, 134)

(c) resolution (118, 16, 68)

(d) resolution (234, 32, 134)

Figure 41 – Skeleton of the plants scene.

SCENE	RESOLUTION	TIME (SECONDS)
Door	(72, 118, 28)	0.17
Door	(144, 234, 54)	1.28
Sponza	(118, 50, 72)	0.92
Sponza	(234, 98, 144)	6.81
Plants	(118, 16, 68)	1.17
Plants	(234, 32, 134)	4.57
Apartment	(234, 128, 152)	11
Apartment	(364, 200, 238)	45

Table 1 – Time required to compute the skeleton.

6.3 SKELETON FILTERING

For some of our method (see Chapters 8 and 11), the number of nodes of the skeleton may be too high since we store and pre-compute a high number of information for each node. By construction, the number of nodes directly depends on the chosen grid resolution (w, h, d).

To obtain a sparser skeleton, we developed a filtering algorithm to adapt the number of nodes to the scene geometry: less nodes in large regions of empty space and more nodes in narrow regions. For that, we use the radius $\mathbf{r}_i$ of the maximal ball centered in each node $\mathbf{n}_i$, by iteratively removing nodes contained in the maximal ball of a larger one, following the topology of the original skeleton graph. The radii can be obtained by computing an euclidean distance map of the digital empty space E and taking the value stored at each node coordinate in this map. Algorithm 4 details the procedure to compute the filtered skeleton. In this algorithm, $\mathcal{N}_{\text{skel}}(\mathbf{n}_i)$ refers to the neighbors of $\mathbf{n}_i$ in the skeleton's graph. Figures 42 and 43 demonstrate the result of our filtering algorithm and compare the number of nodes before and after filtering.

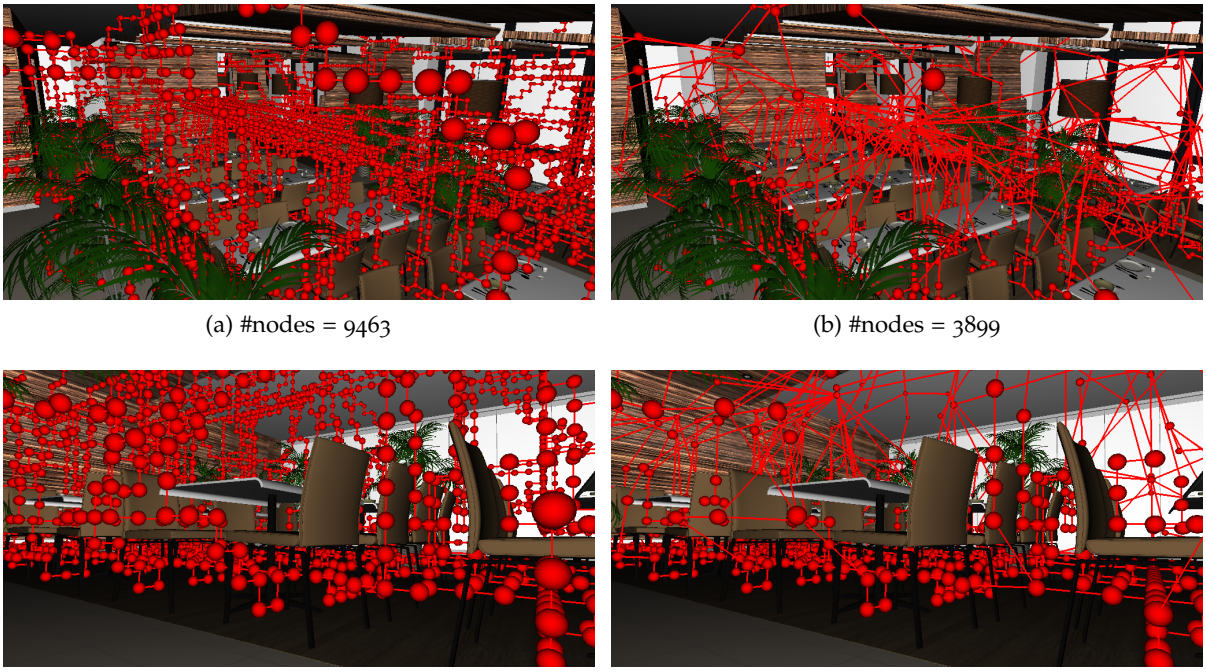
Algorithm 4 : *ComputeFilteredSkeleton*$((\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]})$

Data : A sequence of skeleton nodes $(\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]}$ with their maximal ball radius and neighbors.
Result : The filtered skeleton

foreach $\mathbf{n}_i \in (\mathbf{n}_l)_{l\in[\![1,N_{skel}]\!]}$ **do**
  $P(\mathbf{n}_i) \leftarrow \mathbf{n}_i$
$(\mathbf{n'}_l)_{l\in[\![1,N_{\text{skel}}]\!]} \leftarrow$ sort_largest_radius_first$((\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]})$;
foreach $\mathbf{n}_i \in (\mathbf{n'}_l)_{l\in[\![1,N_{skel}]\!]}$ **do**
  if $P(\mathbf{n}_i) = \mathbf{n}_i$ **then**
    foreach $\mathbf{n}_j \in (\mathbf{n}_l)_{l\in[\![1,N_{skel}]\!]}$ *such that* $\mathbf{n}_j$ *accessible from* $\mathbf{n}_i$ **do**
      if $P(\mathbf{n}_j) = \mathbf{n}_j$ *and* $||\mathbf{n}_j - \mathbf{n}_i|| \leq \mathbf{r}_i$ **then**
        $P(\mathbf{n}_j) \leftarrow \mathbf{n}_i$;
$N \leftarrow \emptyset$;
foreach $\mathbf{n}_i \in (\mathbf{n}_l)_{l\in[\![1,N_{skel}]\!]}$ **do**
  if $P(\mathbf{n}_i) = \mathbf{n}_i$ **then**
    $N \leftarrow N \cup \{\mathbf{n}_i\}$;
    $\mathcal{N}_{\text{filteredskel}}(\mathbf{n}_i) \leftarrow \{\mathbf{n}_j \in \mathcal{N}_{\text{skel}}(\mathbf{n}_i) \mid P(\mathbf{n}_j) = \mathbf{n}_j\}$
foreach $\mathbf{n}_i \in (\mathbf{n}_l)_{l\in[\![1,N_{skel}]\!]}$ **do**
  if $P(\mathbf{n}_i) \neq \mathbf{n}_i$ **then**
    foreach $\mathbf{n}_j \in \mathcal{N}_{skel}(\mathbf{n}_i)$ **do**
      if $P(\mathbf{n}_j) \neq P(\mathbf{n}_i)$ **then**
        $\mathcal{N}_{\text{filteredskel}}(P(\mathbf{n}_i)) \leftarrow \mathcal{N}_{\text{filteredskel}}(P(\mathbf{n}_i)) \cup \{P(\mathbf{n}_j)\}$;
return $(N, \mathcal{N}_{\text{filteredskel}})$;

6.4 SKELETON MAPPING

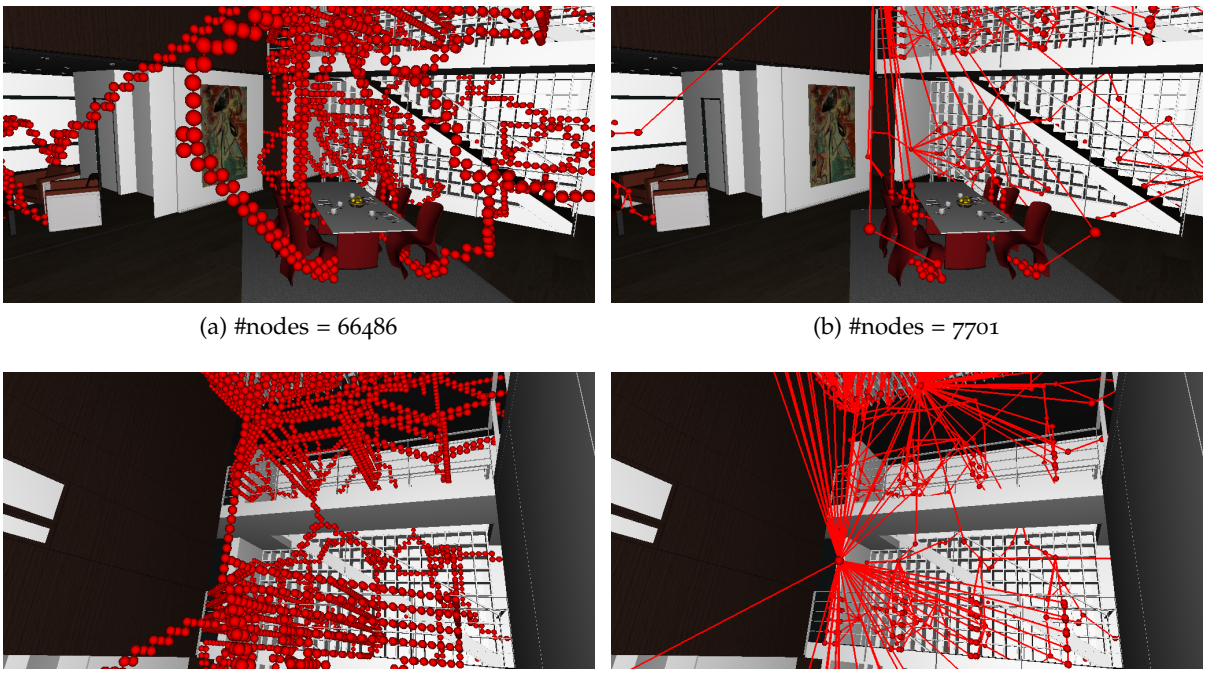
In order to use the skeleton of the empty space in rendering algorithms, we must be able to map each surface point $\mathbf{x} \in \mathcal{M}$ to a skeleton node, denoted $\mathbf{n}_\mathbf{x}$.



(a) #nodes = 9463 (b) #nodes = 3899

(c) #nodes = 9463 (d) #nodes = 3899

Figure 42 – Skeleton (left) and filtered skeleton (right) of the plants scene for a (234, 32, 134) resolution.



(a) #nodes = 66486 (b) #nodes = 7701

(c) #nodes = 66486 (d) #nodes = 7701

Figure 43 – Skeleton (left) and filtered skeleton (right) of the apartment scene for a (364, 200, 238) resolution.

Let $(\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]}$ be the indexed sequence of nodes of the skeleton (either the original skeleton or the filtered skeleton). In order to perform the mapping efficiently (in constant time) and with respect to geodesic distance, we pre-compute a *node mapping grid* G_{skel} that contains a node index at each empty space voxel. This grid is obtained by propagating the index of each node in the digital empty space $E \subseteq G_{w,h,d}$ using the 6-adjacency, as detailed by Algorithm 5. The algorithm simply performs a parallel breadth first search in the grid $G_{w,h,d}$, constrained by voxels covered by the digital empty space E and starting at the grid coordinates $(\mathbf{i}_l, \mathbf{j}_l, \mathbf{k}_l)$ of each node $\mathbf{n}_l$.

Algorithm 5 : *ComputeNodeMappingGrid*$(E, (\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]})$

Data : The digital empty space $E_{\text{D}} \subseteq G_{w,h,d}$. A sequence of nodes $(\mathbf{n}_l)_{l\in[\![1,N_{\text{skel}}]\!]}$ characterized by their grid coordinates and their index.
Result : A grid $G_{\text{skel}} : [\![1,w]\!] \times [\![1,h]\!] \times [\![1,d]\!] \to [\![0,N_{\text{skel}}]\!]$

foreach $(i,j,k) \in [\![1,w]\!] \times [\![1,h]\!] \times [\![1,d]\!]$ **do**
  $G_{\text{skel}}(i,j,k) \leftarrow 0$;
Q $\leftarrow$ FIFOQueue;
foreach $l \in [\![1,N_{skel}]\!]$ **do**
  $G_{\text{skel}}(\mathbf{i}_l, \mathbf{j}_l, \mathbf{k}_l) \leftarrow l$;
  Q.push($\mathbf{i}_l, \mathbf{j}_l, \mathbf{k}_l$);
while *Q not empty* **do**
  $(i,j,k) \leftarrow$ Q.pop();
  foreach $(i',j',k') \in \mathcal{N}_6(i,j,k)$ **do**
    if $(i',j',k') \in E$ *and* $G_{skel}(i',j',k') = 0$ **then**
      $G_{\text{skel}}(i',j',k') \leftarrow G_{\text{skel}}(i,j,k)$;
      Q.push(i',j',k');
return G_{skel};

To associate a surface point $\mathbf{x} \in \mathcal{M}$ with a node $\mathbf{n}_\mathbf{x}$, we first compute the grid coordinates $\mathbf{v}_\mathbf{x}$ of $\mathbf{x}$, obtained from an homogeneous affine transform $T_{\mathcal{M}\to G_{\text{skel}}}$ (this transformation is usually a combination between a translation and a scaling, but may also contains a rotation if the scene is rotated before voxelization):

$$\mathbf{v}_\mathbf{x} := T_{\mathcal{M}\to G_{\text{skel}}}(\mathbf{x}) \tag{151}$$

Since the scene voxelization is the complementary of the digital empty space E_{D} in the grid $G_{w,h,d}$, we have $\mathbf{v}_\mathbf{x} \notin E_{\text{D}}$ and thus $G_{\text{skel}}(\mathbf{v}_\mathbf{x}) = 0$. Consequently, we have to choose a neighboring cell of $\mathbf{v}_\mathbf{x}$ to obtain a skeleton node index stored in the node mapping grid G_{skel}. For that, we use the normal $n_\mathbf{x}$ of the point and an incident direction ω_i to select the correct hemisphere from which the point is viewed. For example, during path sampling (see Section 4.1), the incident direction at a vertex $\mathbf{x}_i$ is the unit vector $\omega_{\mathbf{x}_\text{i},\mathbf{x}_{\text{i}-1}}$ pointing toward the previous vertex of the path. We define $n_\mathbf{x}^+$ as:

$$n_\mathbf{x}^+ := \begin{cases} n_\mathbf{x} & \text{if } n_\mathbf{x} \cdot \omega_\text{i} \geq 0 \\ -n_\mathbf{x} & \text{otherwise.} \end{cases} \tag{152}$$

The direction $n_{\mathbf{x}}^{+}$ points toward the same hemisphere as ω_{i} and we define the normal $n_{\mathbf{x}}^{G}$ in grid space as:

$$n_{\mathbf{x}}^{G} := \frac{T_{\mathcal{M}\to G_{\mathrm{skel}}}(n_{\mathbf{x}}^{+})}{\|T_{\mathcal{M}\to G_{\mathrm{skel}}}(n_{\mathbf{x}}^{+})\|} \tag{153}$$

The empty space voxel $\mathbf{v}_{E,\mathbf{x}}$ associated to $\mathbf{x}$ is then defined as:

$$\mathbf{v}_{E,\mathbf{x}} := \underset{\mathbf{v}\in\mathcal{N}_{26}(\mathbf{v}_{\mathbf{x}})\cap E}{\arg\max}\ (\mathbf{v}-\mathbf{v}_{\mathbf{x}})\cdot n_{\mathbf{x}}^{G} \tag{154}$$

The idea behind this association is to maximize the similarity between the normal direction and the direction pointing toward the neighbor empty space voxel. Finally, the node $\mathbf{n}_{\mathbf{x}}$ mapped to $\mathbf{x}$ is defined by:

$$\mathbf{n}_{\mathbf{x}} := G_{\mathrm{skel}}(\mathbf{v}_{E,\mathbf{x}}) \tag{155}$$

Figure 44 illustrates how a surface point is mapped to a skeleton node, using the described solution. Figures 45 and 46 illustrates skeleton mapping, both for the original skeleton and the filtered skeleton. Every two points with the same color are mapped to the same node of the (filtered) skeleton.

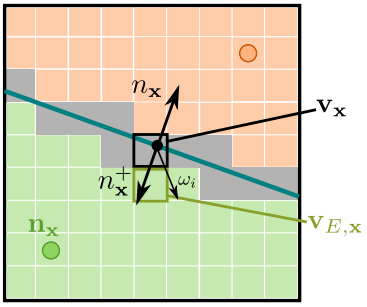
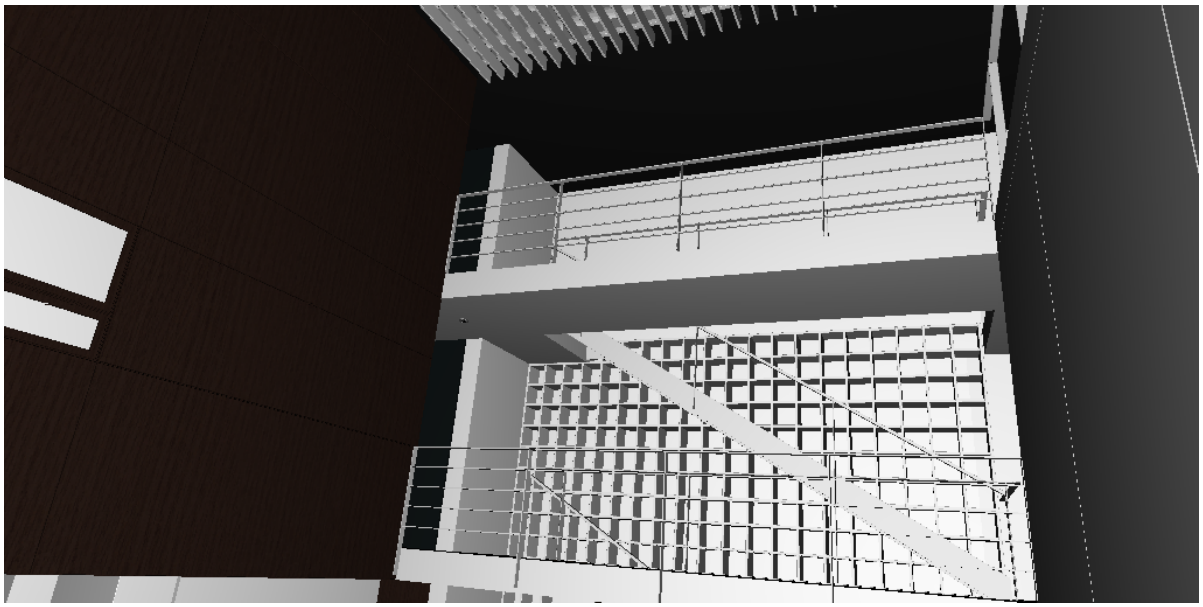
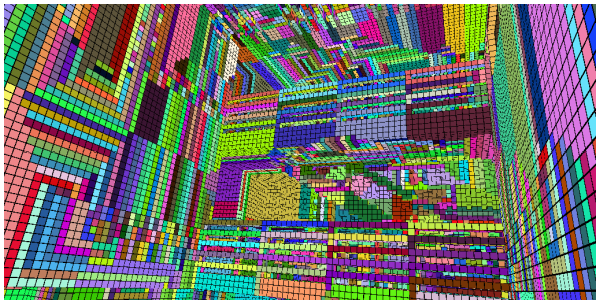


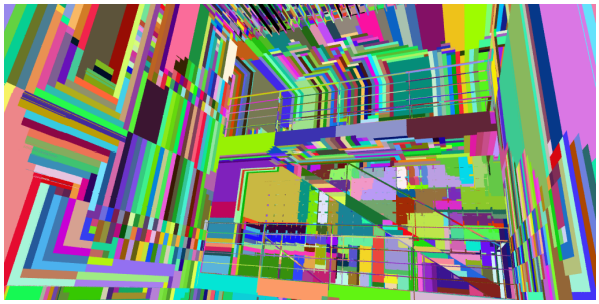
Figure 44 – Illustration, in 2D, of skeleton mapping.



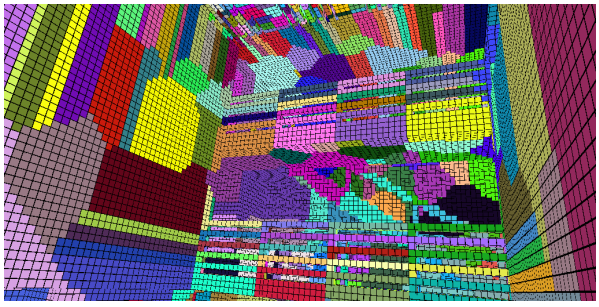
(a) apartment view



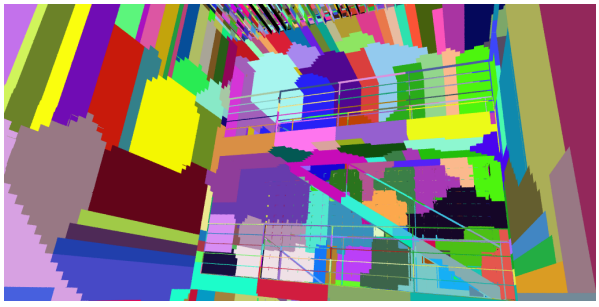
(b) skeleton mapping on voxels



(c) skeleton mapping on surface points

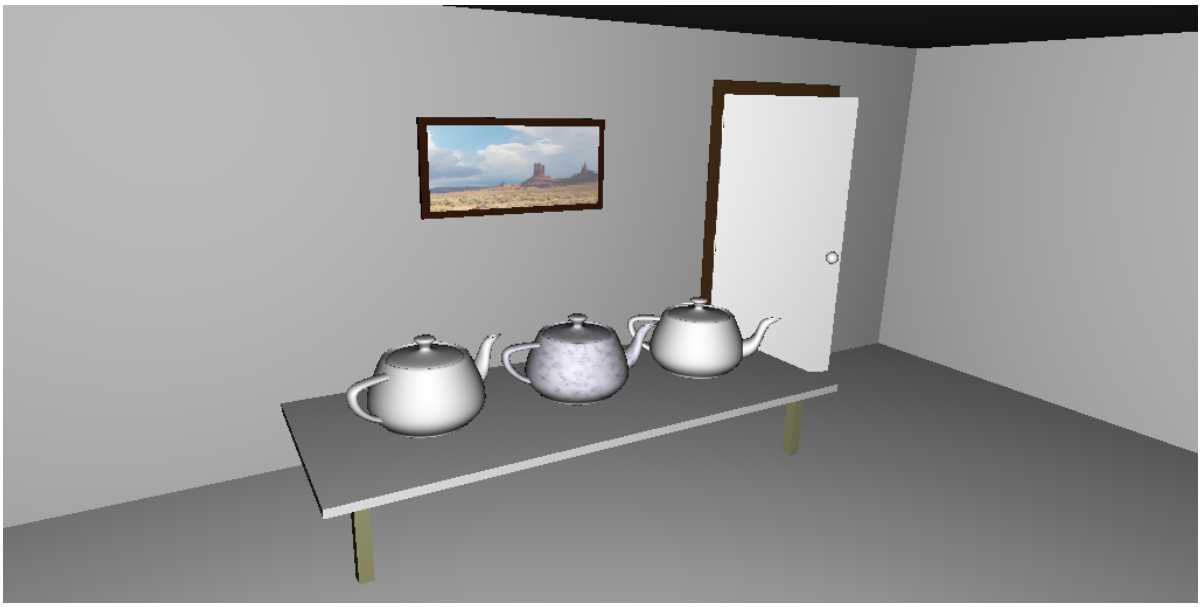


(d) filtered skeleton mapping on voxels

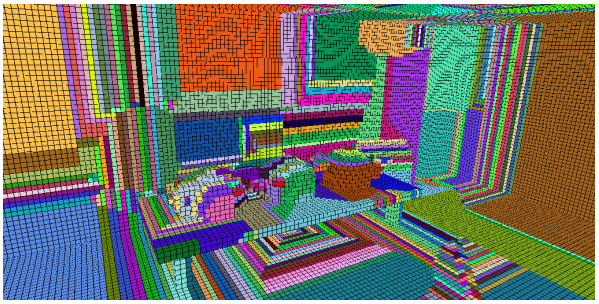


(e) filtered skeleton mapping on surface points

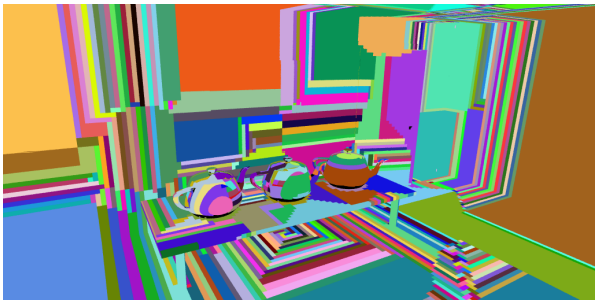
Figure 45 – Skeleton mapping of a view of the apartment scene for a (364, 200, 238) resolution.



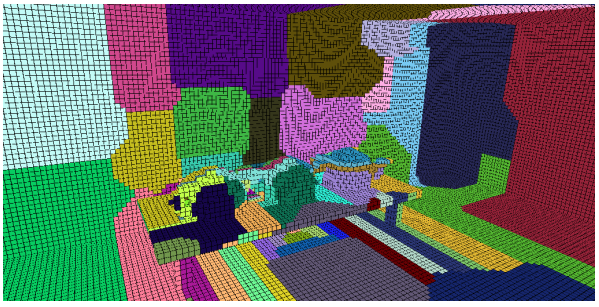
(a) door view



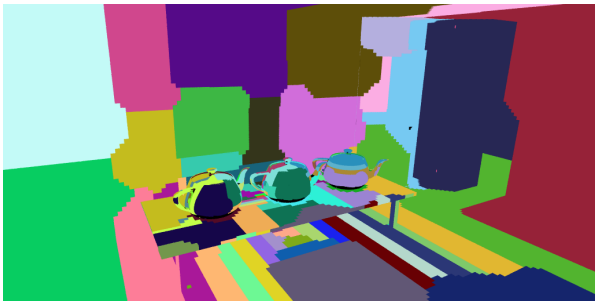
(b) skeleton mapping on voxels



(c) skeleton mapping on surface points



(d) filtered skeleton mapping on voxels



(e) filtered skeleton mapping on surface points

Figure 46 – Skeleton mapping of a view of the door scene for a (144, 234, 54) resolution.

7

REAL-TIME MANY LIGHT RENDERING USING SKELETON VISIBILITY GATES

Interactive 3D applications such as video games have become a lot more realistic this last decade due to the introduction of programmable hardware and the possibility to implement new kinds of algorithms based on global illumination techniques. However, rendering indirect illumination in real-time remains challenging since visibility information between each couple of points (i.e. knowing if they are mutually visible) of the scene must be available. In some configurations, like occluded indoor scenes, indirect illumination becomes the only way to light some regions of the scene (see Figure 47). Many methods have attempted to achieve real time global illumination, but recently a regain of interest has been noticed for methods based on Virtual Point Lights (VPLs) [DKH+14, OPB15], detailed in Section 4.4. Unfortunately, interactive evaluation of the illumination coming from a large set of VPLs remains difficult, especially when the application needs to include shadow computations.

In this chapter, we introduce a new method, presented at the CGAT 2014 conference [NB14a, NB14b], to render a scene illuminated by a large set of VPLs in real-time. Even if our method provides an acceptable frame rate (real-time for up to 512 VPLs), it also produces discontinuity artifacts that are difficult to correct with existing solutions.

Our idea is to approximate visibility using almost convex regions of the empty space of the scene, computed exclusively from the curvilinear skeleton of the empty space. In a given region, we consider all points to be mutually visible, avoiding the need of visibility testing. For visibility between different neighboring regions, we compute *visibility gates*, represented by disks separating regions. These disks are used to convert VPLs of a region into *virtual light cones* emitting light toward a neighbor region. Shadows are therefore rendered without any explicit visibility test, by just associating VPL to fragments according to their respective regions. The association also allows to reduce drastically the number of VPL evaluation per pixel.

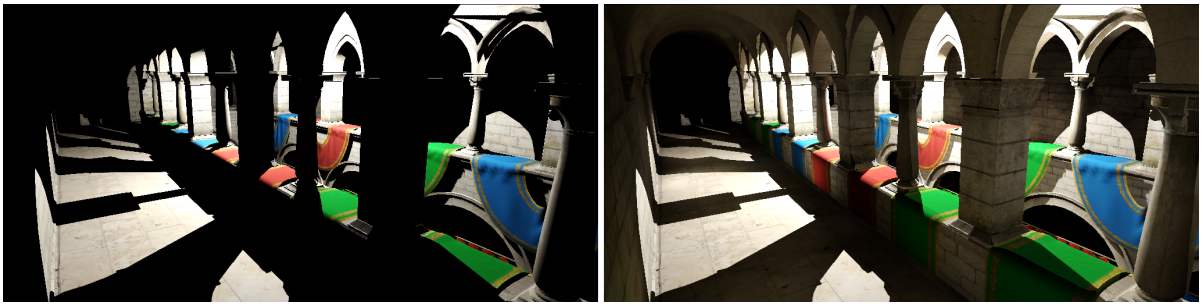


Figure 47 – The left image is a rendering showing only direct illumination. The right image adds indirect illumination, approximated with our method. A large part of this configuration is only illuminated by indirect light, such as the left wall or the back of the pillars.

The discontinuity artifacts we mentioned are actually produced by the segmentation of the scene into discrete regions, illuminated by different sets of VPLs. Despite these artifacts, the illumination computed by the method in each region is coherent when compared to reference images, making it promising for future research. We address and discuss possible future works to solve the discontinuity problem in Section 7.5 of this chapter.

Contents

7.1 OVERVIEW

We begin by a global overview of our method before detailing each step.

The scene is first preprocessed in order to extract a curvilinear skeleton of its empty space. From this skeleton, we compute topological data structures, *visibility clusters* and *visibility gates*, required for rendering. The construction of these structures is detailed in Section 7.2.

The second part of our technique is the rendering loop, that takes advantage of the pre-computed data structure to achieve real-time approximate global illumination. We use a deferred shading rendering pipeline to separate the processing of geometry and lights, and to avoid the illumination of hidden fragments. The steps of the rendering loop are detailed in section 7.3.

Figure 48 illustrates the different steps of our method.

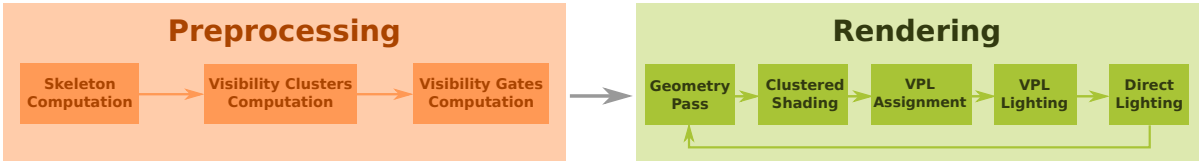


Figure 48 – Overview of our method.

7.2 TOPOLOGICAL CONSTRUCTIONS

Our method is based on a scene clustering computed from a curvilinear skeleton of the empty space of the scene (obtained with the algorithm described in Chapter 6). In this section we detail the preprocessing steps to compute these data. The goal is to extract *visibility clusters* and *visibility gates* from the skeleton.

A visibility cluster is a connected sub-graph of the skeleton's graph such that the union of their maximal balls represents an almost convex region of the empty space. Each visibility cluster is extended to surfaces, using the node mapping grid (described in Section 6.4). Thus, the visibility cluster of a surface point $\mathbf{x}$ is the visibility cluster computed for the node $\mathbf{n_x}$. These structures allow to avoid visibility test between points of a same cluster, since in a convex set any straight line connecting two points is unoccluded. We developed an ad-hoc method to compute such clusters based on the radius of maximal balls and the curvature along the graph of the skeleton. This computation of the visibility clusters is divided in two steps: a simplification of the skeleton to keep few nodes that contains the essential geometrical and topological information; then the clustering applied to the simplified skeleton.

To handle visibility between points belonging to different visibility clusters, we construct *visibility gates*, represented by disks separating neighbor clusters. These disks are used during rendering to build cones of light by combining each VPL of a visibility cluster with each visibility gate associated to the cluster.

7.2.1 *Simplification of the skeleton*

The goal of the simplification is to extract nodes from the original skeleton that represent most of the geometric and topological information. This simplification is similar to the skeleton filtering method described in Section 6.3, but keeps more nodes and classify them according to our needs for the visibility clusters computation.

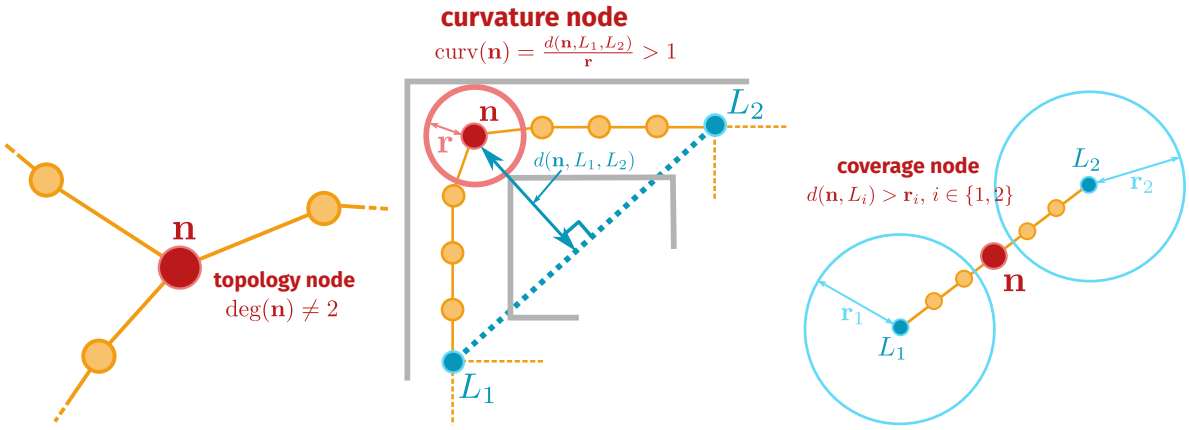
To simplify the skeleton, we classify each node either as *topology node*, *curvature node*, *coverage node* or unclassified. At the end, nodes that are unclassified are removed.

A node $\mathbf{n}$ is marked as *topology node* if its degree $\deg(\mathbf{n})$ (i.e. its number of neighbors) is different than two (see Figure 49a). Such nodes encode connections in the graph and suppressing them can alter the topology of the graph, so we classify them to ensure they are kept in the simplified skeleton.

After this first classification, we regroup nodes of degree two in *skeleton lines*. A skeleton line is a sequence $L = (\mathbf{n}_1, ..., \mathbf{n}_k)$ of nodes such that $\mathbf{n}_i$ is a neighbor of $\mathbf{n}_{i+1}$, $\deg(\mathbf{n}_i) = 2$ and $\mathbf{n}_1$, $\mathbf{n}_k$ both have a topology node as neighbor, respectively noted L_1 and L_2 and called *extrema* of the line L. Such a sequence can be replaced by an edge between L_1 and L_2 without changing the topology of the graph. However, doing so can remove too much geometric information: some edges will pass through walls and important variations in the radius of maximal balls can be lost. Let $\mathbf{n}_i$ be a node of a line L. We define the curvature of n_i by:

$$\text{curv}(\mathbf{n}_i) = \frac{d(\mathbf{n}_i, L_1, L_2)}{\mathbf{r}_i} \tag{156}$$

where $d(\mathbf{n}_i, L_1, L_2)$ is the shortest-distance from $\mathbf{n}_i$ to the straight-line (L_1, L_2) and $\mathbf{r}_i$ is the radius of the maximal ball centered in $\mathbf{n}_i$ and inscribed in the empty space. Intuitively, the higher $\text{curv}(\mathbf{n}_i)$, the farther $\mathbf{n}_i$ is from the straight line (L_1, L_2) in comparison to its distance to the scene. If $\text{curv}(\mathbf{n}_i) > 1$, then the straight-line (L_1, L_2) does not intersect the maximal ball of $\mathbf{r}_i$ and, in that case, replacing L by an edge means replacing a curve by a straight-line, thus inducing a high probability of producing an edge passing through a wall. Consequently,



(a) Topology node (b) Curvature node (c) Coverage node

Figure 49 – Illustration of node classification.

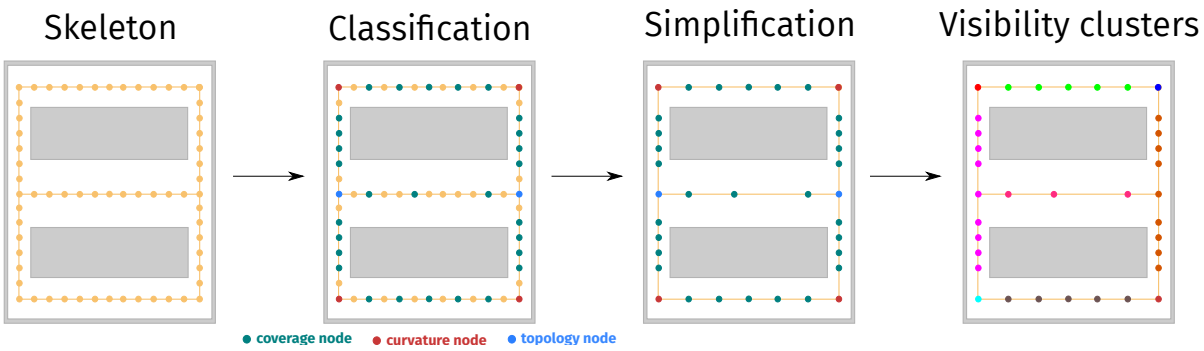


Figure 50 – Simplification of the original skeleton and computation of visibility clusters.

we classify each node with curvature greater than one as a *curvature node* (illustrated by Figure 49b). We build two new lines by setting this node as an new extremum and apply recursively the process until no curvature node can be found on the skeleton lines.

At the end of the process, we obtain a new set of lines such that each extremum is either a curvature node or a topology node. At this point replacing lines with edges can still remove too much geometric information. Indeed, some lines are relatively straight but may contain nodes with many small maximal balls. We want to keep a good coverage of edges of the segmented skeleton by maximal balls. To respect that criterion we check if the nodes of each line are contained in at least one maximal ball centered in the extrema of the line. If a node does not respect that criterion, we classify it as a *coverage node* (Figure 49c). Again we build two new lines by setting this node as an new extremum and apply recursively the process until no coverage node can be found.

Finally we obtain the simplified skeleton by removing all unclassified nodes. The simplified skeleton has a set of nodes that is a subset of the nodes of the original skeleton and each node $\mathbf{n}$ of the original skeleton can be associated with a node $s(\mathbf{n})$ of the segmented simplified by taking the nearest extrema on the line containing that node.

Figure 49 illustrates the process of node classification and Figure 50 illustrates the whole process of skeleton simplification.

7.2.2 *Construction of visibility clusters*

Giving the simplified skeleton, we build *visibility clusters* above it using a greedy ad-hoc algorithm. We developed the algorithm such that the union of maximal balls represented by a cluster is almost convex.

First we sort the nodes of the simplified skeleton according to the radius of their maximal balls, in decreasing order. For each unclustered node $\mathbf{n}$ in that order, we build a new empty visibility cluster $C_{\mathbf{n}}$ and put $\mathbf{n}$ in a queue. While this queue is not empty we pop a node $\mathbf{n}_k$ from it and add it to the cluster $C_{\mathbf{n}}$. We add to the queue all unclustered neighbors of $\mathbf{n}_k$ that are not classified as curvature nodes and for which the radius of the maximal ball does not differ more than α % of the maximal radius contained in the cluster. The parameter α controls how similar two maximal balls must be to cluster together their corresponding nodes. We

used $\alpha = 20$ for our tests, which provide good results in term of convexity. Stopping the propagation at curvature nodes guarantee not to break the convexity of a cluster along a curve of the skeleton (for example at corners of a corridor). We also add to the stack all nodes contained in the maximal ball of $\mathbf{n}_k$ that are accessible from it (by applying a traversal algorithm starting at $\mathbf{n}_k$).

At the end of the process, all nodes are clustered. By construction, each visibility cluster is a connected component of the segmented skeleton. We extend visibility clusters to surface points by associating to $\mathbf{x} \in \mathcal{M}$ the visibility cluster containing $s(\mathbf{n}_\mathbf{x})$, where $\mathbf{n}_\mathbf{x}$ is the node associated to $\mathbf{x}$ by skeleton mapping and $s(\mathbf{n}_\mathbf{x})$ is the node from the simplified skeleton associated to $\mathbf{n}_\mathbf{x}$.

7.2.3 *Visibility gates*

Visibility clusters can be used to avoid visibility testing: if a viewed sample is in the same cluster than a VPL, it is likely that they are mutually visible since clusters are built to approximate convex regions of empty space.

However, a VPL from a cluster generally also illuminate points in other clusters. To approximate efficiently this illumination, we developed the concept of visibility gates, illustrated by Figure 51. For each edge $(\mathbf{n}_i, \mathbf{n}_j)$ of the simplified skeleton such that $\mathbf{n}_i$ and $\mathbf{n}_j$ are in different visibility clusters, we build a disk referred as the visibility gate separating the two clusters. We place the center of that disk at the middle of the edge $(\mathbf{n}_i, \mathbf{n}_j)$, the disk being orthogonal to it and with radius $\min(\mathbf{r}_i, \mathbf{r}_j)$. A subset of light rays leaving each VPLs of the cluster containing $\mathbf{n}_i$ intersect that virtual disk to illuminate points of other clusters. Consequently, we use these visibility gates during the rendering pass by creating for each VPL of a cluster and each visibility gate leaving that cluster, a virtual light cone illuminating points from all other clusters (see Figure 52). Note that by doing so, we completely ignore occlusions that could occur in the cone, which is a reasonable approximation in the context of fast rendering.

7.3 RENDERING

The structures presented in Section 7.2 are used to approximate visibility for indirect illumination, represented by a finite set of VPLs.

VPLs can be computed using path tracing [Kel97, SIMP06a, SIP07] or with rasterization [DS05]. The second method can achieve real-time frame rates for the generation step but limits the number of bounces. Considering the number of VPLs that can be handled in real-time for the illumination step, generating them on the CPU with path tracing is efficient enough and provide more flexibility. Therefore, we chose this solution for our implementation in order to set freely the number of bounces of each light sub-path.

To compute the color $C(\mathbf{z})$ of a view sample $\mathbf{z}$ illuminated by N VPLs $\mathbf{y}_1$, ..., $\mathbf{y}_N$, we apply the following equation, already described in Section 4.4.1, Equation (124):

$$C(\mathbf{z}) = \sum_{i=1}^{N} I(\mathbf{y}_i) M(\mathbf{y}_i, \mathbf{z}) G(\mathbf{y}_i, \mathbf{z}) W(\mathbf{z}) \tag{157}$$

The view sample $\mathbf{z}$ and its sensor response intensity $W(\mathbf{z})$ (often set to one for real-time rendering) is obtained as a fragment generated by rasterization. The intensity $I(\mathbf{y}_i)$ of the VPL and its materials properties are stored while sampling light sub-paths.

The goal of our method is to use the precomputed topological information to quickly approximate the visibility factor $V(\mathbf{y}_i, \mathbf{z})$ that is part of the geometric factor $G(\mathbf{y}_i, \mathbf{z})$. As described in the previous section, we consider two possible cases:

— $\mathbf{y}_i$ and $\mathbf{z}$ are part of the same visibility cluster. In that case we set $V(\mathbf{y}_i, \mathbf{z}) = 1$.

— Otherwise, we consider the N_{gate} visibility gates leaving the cluster containing the VPL $\mathbf{y}_i$ and create N_{gate} virtual light cones with apex $\mathbf{y}_i$ and enclosing the disk of each gate (each cone has infinite extent and can be assimilated to a "spot light"). If $\mathbf{z}$ is outside that cone, we set $V(\mathbf{y}_i, \mathbf{z}) = 0$. Otherwise, we compute a filtered visibility factor $V(\mathbf{y}_i, \mathbf{z}) \in [0, 1]$ that depends on the proximity of the view sample to the border of the cone.

All virtual light cones are created during the generation of VPLs and stored in GPU memory. Testing each view sample against all cones is too costly to be performed in real time. Therefore, we implemented the clustered shading technique [OBA12] that groups view samples in *geometric clusters*, based on their geometric similarity. These clusters allow to quickly test if a group of view samples are outside a virtual light cone.

In this section, we detail each step of the rendering loop. It implements a deferred rendering scheme, but can also be implemented with forward rendering and early depth testing to avoid processing occluded fragments.

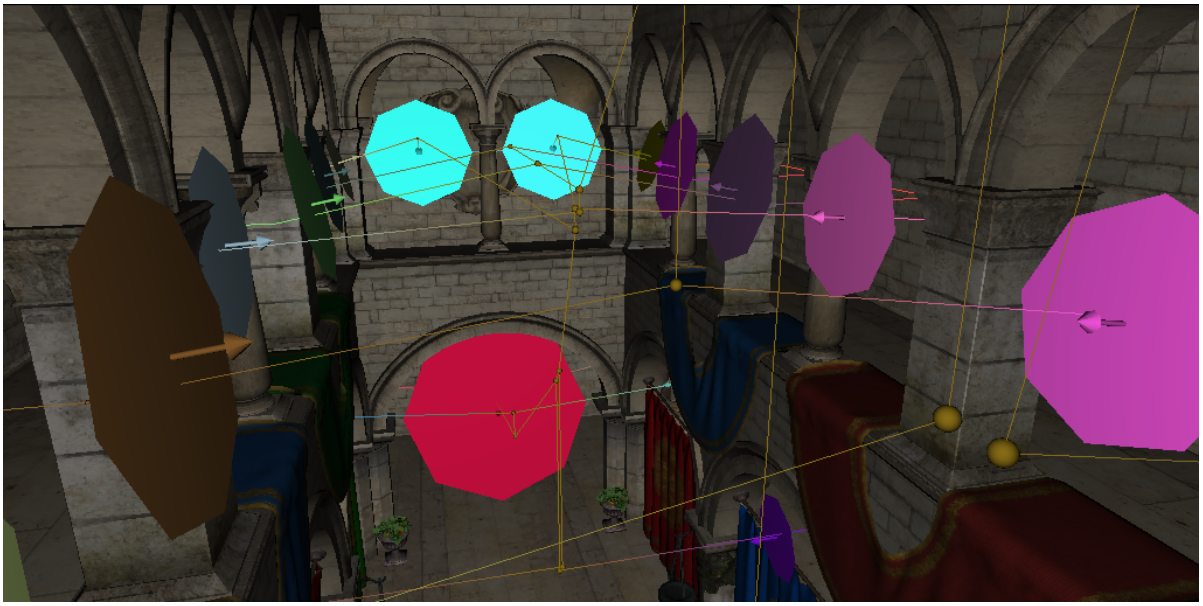


Figure 51 – Illustration of some gates of the sponza scene.

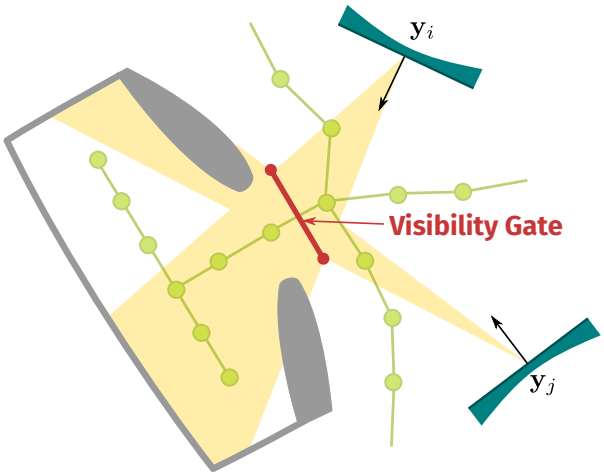


Figure 52 – In this figure, two VPLs are combined with a visibility gate to produce a cone (in 3D) These cones are used to illuminate visibility clusters that do not contain the two VPLs.

7.3.1 *Geometry pass (GP)*

Our geometry pass compute a GBuffer composed of textures storing the following data for each view sample:

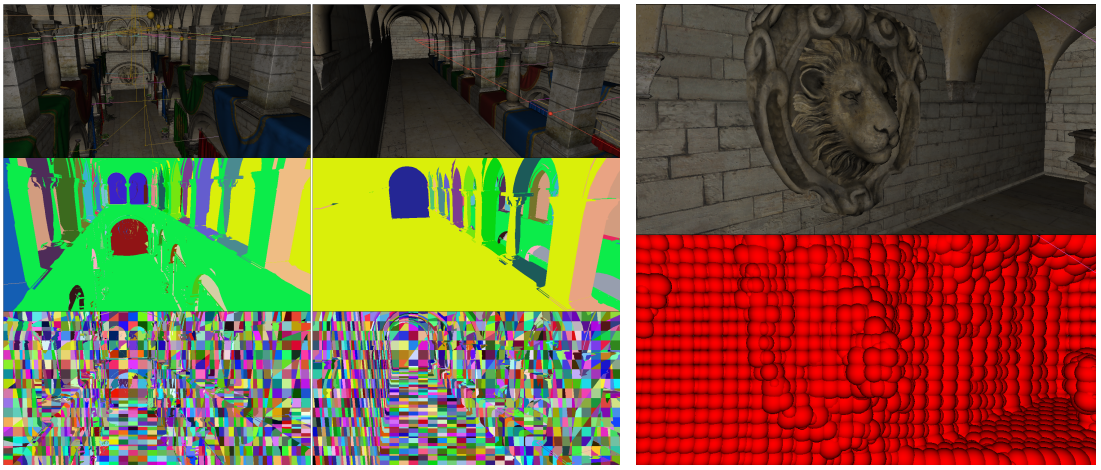
- Normal
- Depth, to reconstruct the position of the view sample
- Diffuse color
- Glossy color and exponent
- Index of the visibility cluster
- Index of the geometry cluster needed for clustered shading

Each pixel of the GBuffer represents a view sample for which Equation (157) has to be evaluated.

To retrieve the index of the visibility cluster of each view sample, we store the node mapping grid in GPU memory and a buffer containing the index of the visibility cluster of each node.

The index of the geometry cluster is obtained according to the clustered shading algorithm [OBA12]: 3D positions inside the view frustum, as well as normal directions are quantized over a finite number of values and the geometry cluster index is obtained from this quantization. Compared to the original method, we add an additional constraint for the geometry clusters: all view samples from a geometry cluster share the same visibility cluster. It makes easier the association of VPLs to geometry clusters, detailed in Section 7.3.3.

Figure 53a shows the index of visibility and geometry cluster of each view sample, represented by false colors.



(a) Visibility clusters (second row) and geometry clusters (third row), represented by false colors.

(b) Bounding spheres of geometry clusters.

Figure 53 – Illustration of geometric and topological data computed during the rendering iteration at each pixel.

7.3.2 *Identification of unique geometry clusters and bounding volumes (IUGC)*

After the computation of the GBuffer, each geometry cluster index is between 0 and the total number of possible index allowed by the quantization. In order to address the geometry clusters efficiently and store their information in a linear buffer, a re-indexing step is necessary (between 0 and the total number of geometry clusters identified in the GBuffer). This computation can be efficiently performed in parallel directly on the GPU, using compute shaders and a local sorting algorithm. For more details about this step, we refer the reader to the article from Olsson et al. [OBA12] describing the original method. Additionally, we compute the bounding box of each geometry cluster, using a parallel reduce min-max on view sample positions of the cluster. These bounding boxes are converted to bounding spheres and used to test efficiently the intersection with virtual light cones during the VPL assignment pass. Figure 53b illustrates the bounding spheres of each geometry cluster for a given view.

7.3.3 *VPL assignement pass (LA)*

For each geometry cluster, this step computes a list containing the index of the VPLs that potentially affects its view samples. Since all view samples stored in a geometry cluster belongs to the same visibility cluster, we add the VPL to the list if it is contained in the same visibility cluster than the geometry cluster (this is the first case of the visibility approximation described earlier). Otherwise, we add the VPL to the list if the bounding sphere of the geometry cluster intersect one of the virtual light cones having the VPL as apex. We also store the index of the visibility gate corresponding to the virtual light cone in order to re-perform the test for individual view samples of the geometry cluster. The VPL assignment pass is implemented with a compute shader, processing all geometry clusters in parallel.

7.3.4 *Indirect lighting (IL)*

We compute the indirect illumination of each view sample using the list of VPLs computed for its geometry cluster. At this point, all VPLs not associated with a visibility gate in the list are considered to be visible from the view sample. For other VPLs, associated with a visibility gate and representing a virtual light cone, we test if the view sample belongs to that cone before adding the contribution. If the VPL is accepted, we multiply the contribution by an attenuation factor based on the cosine between the direction from the VPL to the view sample and the center direction of the cone. This factor is required to attenuate the spot produced by the approximation of the VPL visibility volume with a cone.

7.3.5 *Direct lighting*

This simple step uses primary light sources with shadow maps to add direct illumination in the final image.

7.4 RESULTS AND DISCUSSION

We evaluated the performance of our method on the scene Sponza (Crytek version, 262,267 triangles). All measurements were performed on an NVIDIA GTX 670 Ti GPU. We also exhibit a result computed from the scene Sibenik (75,284 triangles) to discuss the rendering of glossy reflections. The resolution of the frame buffer is set to 1024x512 for all views.

We first present results on view configurations exposing few artifacts in Section 7.4.1. Then in Section 7.4.2 we show and discuss configurations on which discontinuity artifacts are too much important for the algorithm to be usable in its current form.

7.4.1 *Rendering results*

Figures 47, 54, 55 and 56 show results of our method on the Sponza scene from multiple points of view and for different light configurations. All reference images were computed on the CPU using the same set of VPLs and with shadow rays for visibility tests. For each configuration we compare the result of our method with two kind of rendering:

— direct illumination only, to show how indirect illumination improves the realism of the images
— direct illumination + indirect illumination, but without taking into account shadows for VPLs.

Ignoring shadows for indirect illumination is a well-known solution to approximate real-time global illumination. However, as demonstrated by Figure 55, this strategy can strongly overestimate the overall illumination. Our method produces results closer to the reference in terms of illumination. Color bleeding is an effect resulting from the reflection of light coming

Figure 54 – Color bleeding, rendered with our method.

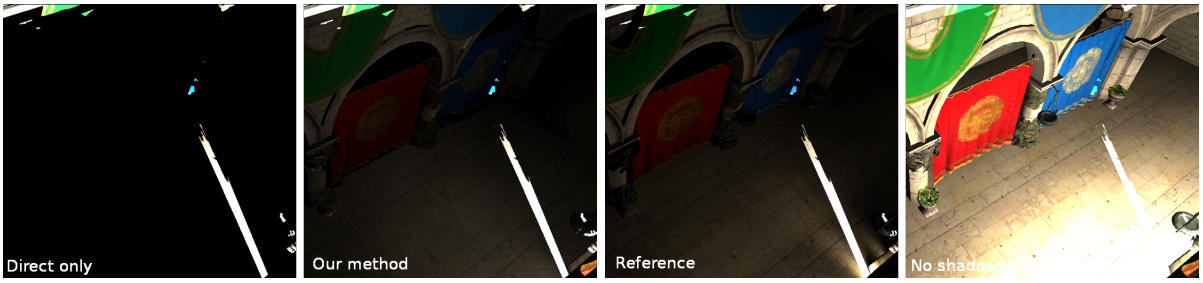


Figure 55 – We compare the result of our method with a rendering that ignore shadows for the virtual point lights. When illumination is mostly indirect, like in this case, doing such an approximation can drastically affects the final image. Our method gives a result close to the reference thanks to the subset of VPLs well-chosen by our algorithm to illuminate each view sample.

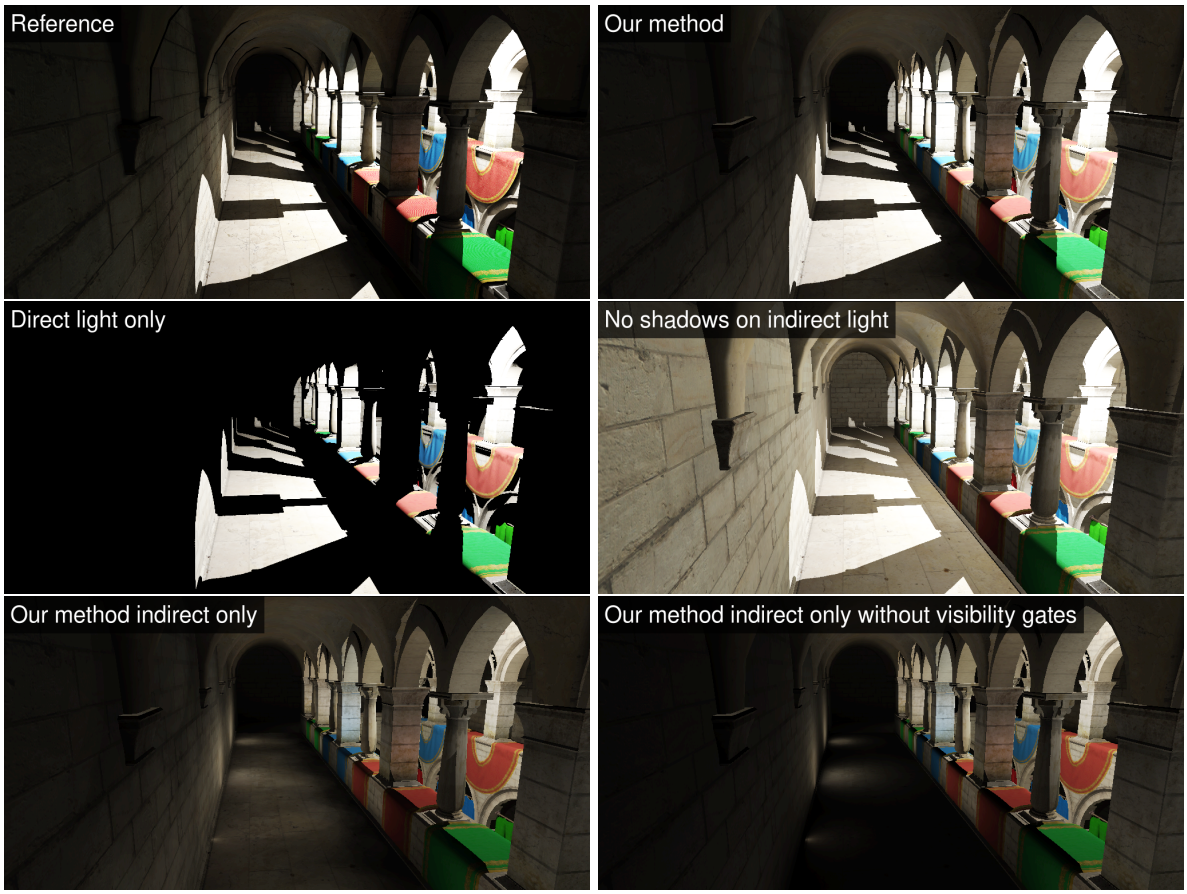


Figure 56 – Here we show how visibility gates can affect the indirect illumination. Most of the light on the floor is produced by virtual point lights that are not in the same cluster. By using visibility gates, we can find for each view sample a subset of these virtual points lights that certainly illuminate it. Since gates are only disks, we still miss a lot of VPLs, producing darker results than the reference in some part of the result. Other parts are over evaluated due to shadow tests that are ignored inside visibility clusters. Still, our result is a better match of the reference image than when we totally ignore shadows for virtual point lights.

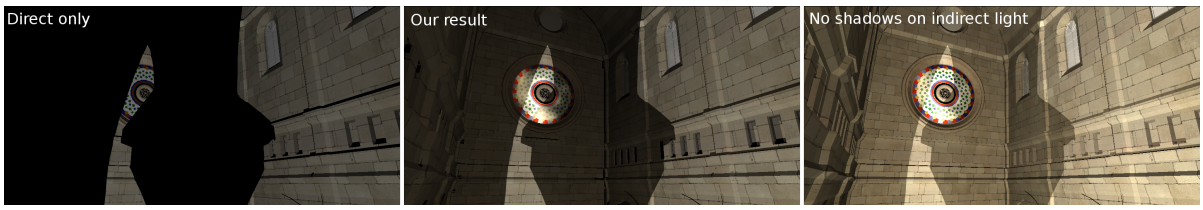


Figure 57 – This view of Sibenik church demonstrates glossy reflections on the stained glass. A high number of VPLs must be used to get correct glossy reflections (here 4192 VPLs were used). There exists other methods which are more appropriated to compute that kind of effect.

VPL #	GP	IUGC	LA	IL	TOTAL
512	6.1	4.3	12.3	10.9	33.6
1024	6.1	4.3	24.5	22.1	57
2048	6.1	4.3	49.0	44.2	103.6

Table 2 – Rendering Times (ms)

from close colored textures. It can be observed rendered by our method in Figures 54 and in the indirect illumination snapshot of Figure 56.

Glossy reflections are generally badly estimated by methods based on VPLs. Indeed, using a discrete set of points to approximate incident indirect illumination does not permit the evaluation of incident radiance along directions having a high importance for the glossy BSDF. Figure 57 shows a view of the Sibenik scene where glossy reflections can be observed on the stained glass, with 4192 sampled VPLs. With that number of VPLs, our method is no longer real-time but remains interactive (3.7 frames per second in that case). A more appropriate method to render glossy reflections in real-time is [CNS+11] but is limited to two bounces of light.

Table 2 gives times in milliseconds of each step of the rendering pass for different numbers of VPLs. We achieve real-time frame rate for a number of VPLs less than 512 and stay interactive for few thousands of VPLs.

Table 3 presents mean square errors (MSE) computed for two different configurations. These MSE have been computed on color values in the range $[0, 255]$. The table shows a lower error for our method than for direct lighting or indirect lighting without indirect shadows. Configuration 1 is the one illustrated in Figure 56 and Configuration 2 is the one shown in Figure 55. Since there is few indirect illumination on this configuration (compared to direct lighting), the error produced by computing direct lighting only is closed to the one produced by our result.

We implemented the light assignment pass using a brute force algorithm on GPU (each geometry cluster is tested in parallel against each virtual cone light). This pass can be optimized using a acceleration data structures on lights.

The performance of our algorithm depends on the lighting and viewing configuration: if all VPLs are in the same visibility cluster, then rendering takes much longer if the camera look at a many points from that cluster. The rendering times given in the table are obtained with the view point and primary light source position of Figure 56. Finally, it can also be a good solution to remove lighting using visibility gates. That way, we just keep indirect illumination that occur inside visibility clusters. This solution loses some long range lighting but also improve the performances of the algorithm by removing small contributions.

CONFIG.	DIRECT LIGHT	NO SHADOWS ON INDIRECT	OUR METHOD
1	111.96	69.1091	33.1041
2	39.7544	170.121	39.6128

Table 3 – Mean Square Error

7.4.2 *Limitations*

7.4.2.1 *Discontinuity artifacts*

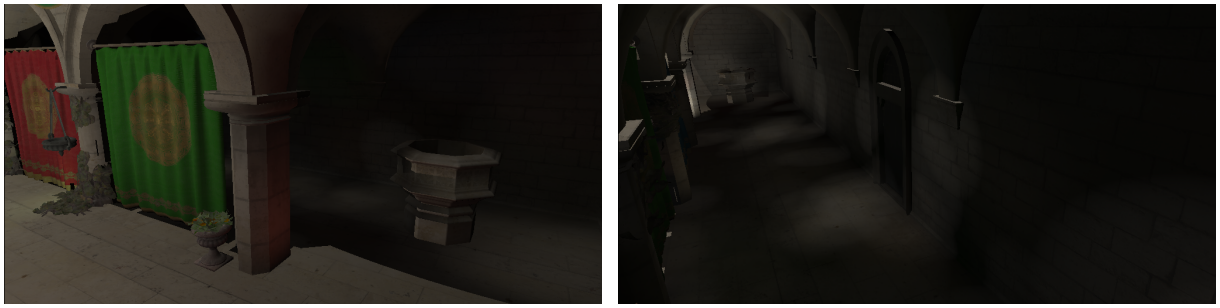
As discussed in the introduction of this chapter, our method is subject to the presence of discontinuity artifacts in the rendering of indirect illumination (Figure 58a). These artifacts are caused by two possible sources:

— A view sample is associated to a bad visibility cluster because of the node mapping grid.
— The discrete segmentation of view samples in visibility clusters produces fast change in the indirect illumination at places where this variation should be smooth.

The first source of artifacts is due to the way we compute the node mapping grid. While being correct, most of the time, in term of distance, visibility and orientation, the geodesic propagation of node indices produces some incoherences at some voxels, for which the associated node differs strongly from neighbor voxels. While the impact of this problem is limited for off-line rendering algorithms based on random sampling, thanks to multiple importance sampling (see Chapter 9 and Chapter **??**), it is hard to correct for real-time algorithms where a decision is made based on the node associated to each view sample. A possible solution to address this problem is to te replace the associated node of a surface point "on the fly" with a neighbor node in the graph, using a similarity measure well defined between surface points and nodes. We tried various similarity measures, using surface orientation, distance and maximal ball radii, but failed to find one that fixes all bad associations without introducing new ones. We think that the correct way of fixing this issue is to take the problem at its source: the construction of the node mapping grid. All contributions presented in this thesis would benefit from a good propagation algorithm that associate nodes to voxels according to geometric properties of surfaces and visibility in the scene.

The second source of artifacts is related to the method itself, that partition view samples in discrete regions according to their associated nodes. Applying such kind of partitioning on view samples and computing their illumination according to it is actually quite dangerous since they are directly seen by the viewer. While it is easy to detect where discontinuities occur in the final image (by analyzing the visibility cluster image), smoothing correctly the computed illumination is a hard task since the size and geometry of visibility clusters vary in a non predictable way (as well as the illumination variation).

These discontinuity issues led us to develop another real-time rendering algorithm addressing the same problem, but avoiding any kind of partitioning on the view samples. This second method is presented in Chapter 8 and renders a convincing approximation of shadows from a large set of VPLs with no discontinuity artifacts.



(a) Discontinuities produced by our method.

(b) Insufficiency of disks to render the indirect illumination between visibility clusters.

Figure 58 – Illustration of rendering artifacts generated by our method.

7.4.2.2 *Visibility gates limitations*

Our visibility gates are represented by disks, which is a coarse approximation of the empty space interface between two visibility clusters. While the idea of replacing visibility testing by geometrical proxies is interesting and has already been used in production rendering application (these proxies are usually positioned manually and not automatically extracted), they must adapt well to the geometry of the scene. Using our skeleton to extract visibility gates is not enough to achieve that goal since every node only stores the radius of its maximal ball, which correspond to its distance to the scene. Therefore, no information about the main orientation and extent of the empty space is available at each node and we are limited to disks to represent the interface (Figure 58b).

To address this limitation, Chapter 10 presents a more robust method to extract portals from empty space, using an opening map instead of a skeleton. While we developed this other method for an application to ray sampling, these portals can be used to replace visibility gates and to provide better accuracy in the estimation of visibility.

7.5 CONCLUSION AND PERSPECTIVES

We presented a new real-time rendering algorithm that approximate visibility on indirect illumination using a clustering strategy combined with geometrical proxies in the form of disks. While this method produces a coherent illumination in each cluster, the presence of discontinuity artifacts between clusters and the disk representation of visibility gates make the result too coarse to be used in practice for production rendering.

Despite the time spent on this method without providing, at the end, artifacts-free images, we believe that facing these issues allowed us to better understand the limitations brought by the skeleton and how to deal with them. While the skeleton is a powerful tool, bringing many information about the scene and its visibility, the discrete segmentation of the surfaces of the scene according to nodes has to be carefully exploited in order to avoid artifacts in rendered images. We must also remain aware that the geometric information carried by the skeleton is somewhat limited, at least for a curvilinear one. The maximal balls centered in each node

gives us a lower bound on the local volume of empty space occupied at each node but cannot describe its orientation and extent in directions orthogonal to its graph. Consequently, either we have to use another structure to retrieve that information, either we have to use it in a conservative way, that is without taking the risk of reducing the rendering quality of images.

Possible future works related to the ideas developed for this method would address the following points:

— Improving the robustness of visibility clusters in terms of convexity to reduce errors made by approximating the visibility inside of them. Also, to avoid discontinuity artifacts, defining the belonging of a view sample to a visibility cluster as a function taking values in the interval $[0, 1]$ instead of a binary value. We already developed some experiments toward that goal that seem promising.

— As mentioned previously, replacing the representation of visibility gates by meshs more adapted to the local geometry of the empty space. However, an issue that could arise in the context of real-time rendering would be the cost associated to the evaluation of the visibility between a VPL and a view sample. The advantage of disks is that a VPL combined with a disk forms a virtual cone, that is a simple geometrical shape to test for intersection with bounding volumes of view samples.

8

SKELETON SHADOW MAPPING

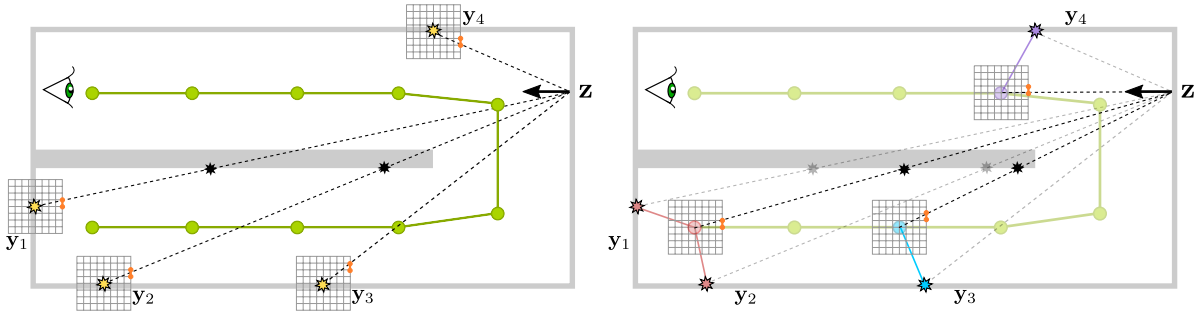
This chapter presents a second real time method for many-light rendering, with the same purpose as Chapter 7, the approximation of shadows using the skeleton. This new approach is based on shadow mapping and we do not partition view samples according to the skeleton, in order to avoid any discontinuity artifact. Our results demonstrate that the method effectively renders a smooth and convincing indirect illumination, with better frame rates than our previous one.

However, the work presented in this chapter is still in progress and presents some limitations. For example, our current implementation is limited to diffuse scattering, while the method in itself can also be applied to glossy reflexions. Some optimizations also need to be implemented, such as interleaved shading, to compare our method to concurrent recent works dealing with real-time indirect illumination from many-lights.

Our goal is not to provide a correct physical approximation of indirect illumination, but an efficient and plausible rendering of indirect shadows that can be used to trick the viewer in believing that they are real. For that purpose, the approximated indirect illumination intensity must match the real one, and approximated shadows must appear at locations that make sense geometrically. Our rendering technique targets applications that require fast and convincing rendering, such as video games or virtual reality applications.

We give an overview of the method in Section 8.1, as well as some motivations regarding our idea. A mathematical formulation of our approximation is provided in Section 8.2. Section 8.3 details the preprocessing of data required by the method, a shadow map for each node. Each important step of the rendering loop is then explained in Section 8.4. We demonstrate and discuss our results in Section 8.5. Finally, we conclude and discuss some perspectives for the works presented in this chapter in Section 8.6.

Contents



(a) Standard shadow mapping (b) Skeleton shadow mapping

Figure 59 – Illustration of the visibility factorization performed by our method. We observe in this simple example that our strategy gives correct results for VPLs $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_4$, but fails for VPL $\mathbf{y}_3$.

8.1 OVERVIEW AND MOTIVATION

We observed in Chapter 7 that partitioning view samples according to the skeleton mapping function produces discontinuity artifacts that need to be filtered. A good filtering strategy is however difficult to develop since surface regions created by the skeleton mapping are quite extended in space. Consequently, we decided to avoid clustering of view samples for the method presented in this chapter.

Our idea is to factor visibility between VPLs associated to the same node. To achieve that goal, we pre-compute a shadow map for each node, centered at its position, before the rendering loop. During rendering, for a VPL located at position $\mathbf{y} \in \mathcal{M}$, we use the shadow map computed for the node $\mathbf{n}_\mathbf{y}$. This strategy is illustrated by Figure 59. There is two main advantages of using such visibility factorization:

— Since a large number of VPLs are likely to be assigned to a given node $\mathbf{n}$, we can avoid their processing for each view sample not visible from $\mathbf{n}$ and thus achieve high performance.

— The shadow maps do not have to be re-computed each frame even if the VPLs are dynamic. The drawback behind this is that the geometry of the scene have to remain static. Indirect shadows produced by our method also remains static, in terms of positioning.

The purpose of the work presented in this chapter is to evaluate experimentally the viability of this approximation for rendering non realistic but plausible indirect shadows. Targeted applications are those that do not require precise physical results but convincing rendering, such as video games or virtual reality simulations. While our approximation might seems too coarse at first glance, since skeleton nodes are centered in the empty space and not located on surfaces, our results demonstrate that it works well in practice and does not produce visible artifacts when combined with standard rendering techniques, such as *percentage closest filtering* (PCF) to obtain soft shadows.

8.2 MATHEMATICAL FORMULATION

Recall the equation to compute the color $C(\mathbf{z})$ of a view sample $\mathbf{z}$ illuminated by a set $Y = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ of VPLs (given in Section 4.4):

$$C(\mathbf{z}) = \sum_{i=1}^{N} I(\mathbf{y}_i) M(\mathbf{y}_i, \mathbf{z}) V(\mathbf{y}_i, \mathbf{z}) G^*(\mathbf{y}_i, \mathbf{z}) W(\mathbf{z}) \tag{158}$$

where we extracted the visibility factor from the geometry factor, such that the identity $G(\mathbf{y}_i, \mathbf{z}) = V(\mathbf{y}_i, \mathbf{z}) G^*(\mathbf{y}_i, \mathbf{z})$ holds. Our method approximates Equation (158) with the following expression:

$$\widehat{C}(\mathbf{z}) = \sum_{i=1}^{N} I(\mathbf{y}_i) M(\mathbf{y}_i, \mathbf{z}) V_f(\mathbf{n}_{\mathbf{y}_i}, \mathbf{z}) G^*(\mathbf{y}_i, \mathbf{z}) W(\mathbf{z}) \tag{159}$$

where the visibility $V(\mathbf{y}_i, \mathbf{z})$ between the VPL and the view sample has been replaced with a filtered visibility factor $V_f(\mathbf{n}_{\mathbf{y}_i}, \mathbf{z}) \in [0, 1]$, between the node associated to the VPL and the view sample. The filtering of visibility is essential here to obtain soft shadows, that provide a better approximation of indirect shadows.

Let $Y_{\mathbf{n}_i}$ be the set of VPLs associated to the i-th node $\mathbf{n}_i$ of skeleton, defined as:

$$Y_{\mathbf{n}_i} \coloneqq \{\mathbf{y} \in Y \mid \mathbf{n}_\mathbf{y} = \mathbf{n}_i\} \tag{160}$$

Equation (159) can be rewritten by summing over the M nodes of the skeleton, and factoring the visibility factor:

$$\begin{aligned} \widehat{C}(\mathbf{z}) &= \sum_{i=1}^{M} V_f(\mathbf{n}_i, \mathbf{z}) \sum_{\mathbf{y} \in Y_{\mathbf{n}_i}} I(\mathbf{y}) M(\mathbf{y}, \mathbf{z}) G^*(\mathbf{y}, \mathbf{z}) W(\mathbf{z}) \\ &= \sum_{i=1}^{M} V_f(\mathbf{n}_i, \mathbf{z}) C_{Y_{\mathbf{n}_i}}(\mathbf{z}) \end{aligned} \tag{161}$$

The computation of the inner sum, expressed as $C_{Y_{\mathbf{n}_i}}(\mathbf{z})$, requires to loop over all VPLs associated to the node $\mathbf{n}_i$. This computation can be immediately discarded if the node does not see the view sample, that is if $V_f(\mathbf{n}_i, \mathbf{z}) = 0$. Consequently, our approximation allows to avoid many computations for each view sample.

8.3 SHADOW MAPS COMPUTATION

Our method approximates the visibility function of many VPLs using a shadow map pre-computed for each node. Since the storage of these shadow maps can be expensive, we apply the method on the *filtered skeleton*, detailed in Section 6.3, which contains less nodes than the original skeleton and adapts the node density to the local geometry of the empty space.

For each node $\mathbf{n}$ of the filtered skeleton, we pre-compute a *cube shadow map*. All shadow maps are stored in several *cubemap array textures*, such that many shadow maps can be computed in a single draw call using instanced rendering and geometry shaders.

We discuss the resolution we use for the cubemaps in the result section, as well as possible optimizations to reduce memory consumption when the filtered skeleton still contains too many nodes.

8.4 RENDERING

Our method implements a deferred rendering loop. The geometry pass computes a GBuffer containing for each pixel its normal, its depth and its material properties.

The set of VPLs approximating indirect illumination is sampled with standard solutions [Kel97, DS05] at the beginning of the rendering loop. Each VPL is then associated to a node using skeleton mapping.

The following sections detail the specific steps of our method to approximate indirect illumination, by computing Equation (161) for each pixel of the image, in parallel thanks to the GPU implementation. For that, we iterate over each skeleton node $\mathbf{n}_i$ associated to at least one VPL ($|Y_{\mathbf{n}_i}| \neq 0$), in order to accumulate each term $V_f(\mathbf{n}_i, \mathbf{z})C_{Y_{\mathbf{n}_i}}(\mathbf{z})$ of Equation (161).

The complete rendering loop is summarized as follows:

- Compute GBuffer
- Sample a set $Y = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ of N VPLs
- For each VPL $\mathbf{y}_i$:
  - $Y_{\mathbf{n}_{\mathbf{y}_i}} \leftarrow Y_{\mathbf{n}_{\mathbf{y}_i}} \cup \{\mathbf{y}_i\}$
- For each node $\mathbf{n}_i$ such that $|Y_{\mathbf{n}_i}| \neq 0$:
  - Compute shadow framebuffer of $\mathbf{n}_i$ (SFC)
  - Accumulate contributions of VPLs from $Y_{\mathbf{n}_i}$ (CA)
- Accumulate direct illumination

The two steps highlighted in red are the one specific for our method and are described in the two next sections.

8.4.1 *Shadow framebuffer computation (SFC)*

We first compute a *shadow framebuffer* for the node $\mathbf{n}_i$ of the current iteration. The purpose of this framebuffer is to contain, for each pixel j, the filtered visibility factor $V_f(\mathbf{n}_i, \mathbf{z}_j)$, where $\mathbf{z}_j$ is the view sample of the pixel. This factor is obtained from the shadow map precomputed for the node $\mathbf{n}_i$, using *percentage closer filtering* (PCF) [RSC87] to handle the filtering of the visibility function of the node. This technique provides both anti-aliased shadows (aliasing being a well know problem of shadow mapping) and soft shadows approximation.

8.4.2 *Contribution accumulation (CA)*

The second step accumulates the contribution coming from VPLs of the set $Y_{\mathbf{n}_i}$. The fragment shader reads the value $V_f(\mathbf{n}_i, \mathbf{z}_j)$ stored in the shadow framebuffer and skip the computation

if $V_f(\mathbf{n}_i, \mathbf{z}_j) = 0$. Otherwise, the sum of individual contributions of VPLs from $Y_{\mathbf{n}_i}$ is accumulated and scaled by $V_f(\mathbf{n}_i, \mathbf{z}_j)$.

8.5 RESULTS AND DISCUSSION

We compare our indirect shadows approximation against a computation of exact shadows, obtained with CPU rendering using the same set of VPLs. We also show, for each result, a rendering for which shadows on indirect lighting are ignored, in order to demonstrate that they are essential for visual realism when indirect lighting constitute the main contribution for the illumination of the image. For now, we are mainly interested in visual comparisons rather that statistical ones (with error measurement), since our method provides a coarse approximation of indirect shadows. We compare and discuss differences between illumination intensity and shadow positioning, and we keep for future works the development of an adapted and systematic comparison approach to validate our results.

All presented images where rendered with a NVidia GeForce GTX 670 GPU for real-time rendering, and an Intel Core i7-3770K CPU 3.5GHz for offline CPU rendering. Rendering times, for indirect illumination only, are given by Table 4 for various number of VPLs, corresponding to 500 light paths, 1000 light paths and 2000 light paths, each light path being limited to one edge (one bounce indirect illumination).

All displayed images are the one obtained with the highest number of VPLs recorded by the table, for each configuration. In rendered images, some surfaces are black because of a specular material assigned to the surface (either reflective or transmissive), and our implementation has not been adapted to these kind of materials.

8.5.1 *Visual comparisons*

Figure 60 to Figure 66 illustrate visual comparisons between an exact rendering of shadows, computed on the CPU, our method and a rendering featuring no shadows on indirect lighting, both computed on the GPU. Overall, our results demonstrate a good approximation of indirect illumination, which is more close visually to the exact rendering than ignoring shadows. Globally, our method either overestimate or underestimate a little bit the illumination, depending on the distribution of VPLs among skeleton nodes that are visible from view samples. For the viewpoint illustrated by Figure 62, our result does not match well the CPU image because of many fake shadows produced at the ground. Nevertheless, these shadows are coherent with the scene geometry and they do not harm the plausibility of the result since the viewer is not aware that they do not exist in a physical rendering. An important problem, however, is the coherency between direct and indirect shadows, especially for dynamic lighting. By construction, our method renders static shadows, since skeleton nodes are static in the scene. In a dynamic setting for light sources, it results in moving shadows for direct lighting and static for indirect lighting, which can break the feeling of immersion of the user. This problem can also be noticed on static images, such as Figure 63, when the direct illumination part on the image does not correctly align with indirect shadows.

8.5.2 *Rendering times*

Table 4 records rendering times for our method, expressed in milliseconds. We achieve real time frame rate is achieved for about 1000 VPLs, and remain interactive for approximately 2000 VPLs. The performance of our method depends on the viewpoint, since the accumulation of VPL contributions depends on the visibility between view samples and skeleton nodes. The number of shadow framebuffer computation (SFC) pass depends on the distribution of VPLs. If VPLs are scattered on many skeleton nodes, more SFC passes have to be performed and the performance can drop. Overall, the most costly part is the contribution accumulation (CA), since it depends on the number of VPLs. Recent works on many-light real time rendering [BBH13, Tok14] report real-time frame rate for approximately 1024 VPLs, with shadows computed with the imperfect shadow map (ISM) technique [RGK+08, BBH13]. However, these methods make use of the interleaved shading solution [KH01, SIMP06b], that distribute the number of VPL contribution computation on tiles, and reconstruct an approximation of the complete sum of contributions through filtering at each pixel. For example, the method presented in [Tok14] uses 8×8 tiles, and then actually process $1024/(8 \times 8) = 16$ virtual lights per pixel. Our method provides real-time frame rate for approximately the same number of VPLs, without using interleaved shading, which make it promising to handle thousands of VPLs in real-time after the implementation of this strategy.

8.5.3 *Memory consumption and shadow maps resolution*

In our implementation, we used cubemaps with a resolution of 6×128^2 pixels, containing depth values represented as floats. With this setting, each shadow map consumes approximatively $400kB$, allowing about 2500 shadow maps to be stored in a $1GB$ GPU memory.

If the filtered skeleton still contains too many nodes for storing all shadow maps on GPU, they can be converted to 2D shadow maps using spherical mapping or dual paraboloid mapping [BApS02].

8.6 CONCLUSION AND PERSPECTIVES

We proposed a new method for the approximation of indirect illumination using VPLs and the filtered skeleton of the empty space of the scene. While this work is in progress, our results are promising for real-time rendering of convincing shadows, in order to provide fast approximation of indirect illumination for entertainment applications.

Future works will be focused on the implementation of missing features, such as glossy scattering and interleaved shading. Regarding glossy scattering, a recent method from Tokuyoshi [Tok14] proposed to use Virtual Gaussian Spherical Lights (VGSLs) instead of VPLs. VGSLs gives a better approximation of glossy reflections, and we plan to adapt our method for the use of such many-light representation.

CONFIG.	# VPLS	SFC	CA	TOTAL	NOSHADOWS CPU (SEC.)	
Sponza1	484	4	8.7	**12.7**	19	67
	961	6.5	17.6	**24.1**	40	128
	1936	9.2	34.6	**43.8**	77.3	232
Sponza2	464	10.5	11.5	**22**	20	67
	924	17.1	22.5	**39.6**	40	154
	1867	24.3	42.8	**67.1**	78	300
Sponza3	484	4.5	9.5	**14**	19.7	91
	961	5.9	18.5	**24.4**	38	169
	1936	8.5	36.9	**45.4**	77	328
Sponza4	481	4	3.1	**7.1**	19.3	50
	954	6	6	**12**	38.5	103
	1921	7.1	10.3	**17.4**	76.3	200
Door	484	5	10.3	**15.3**	20	50
	961	5.6	20	**25.6**	38.7	107
	1936	5.6	39	**76.4**	76.4	203
Apartment1	484	12	11.7	**23.7**	22.4	82
	961	18	20	**38**	40	152
	1936	19.1	36	**55.1**	77.5	291
Apartment2	484	9.8	12.4	**22.2**	22.4	82
	961	11.5	22.5	**34**	40	152
	1936	17.8	42.6	**60.4**	77.5	291

Table 4 – Rendering times, all expressed in milliseconds except the CPU image computation time expressed in seconds. Timings for the column "total" correspond to our method. The column "noshadows" records timings for the computation of indirect illumination without shadows.

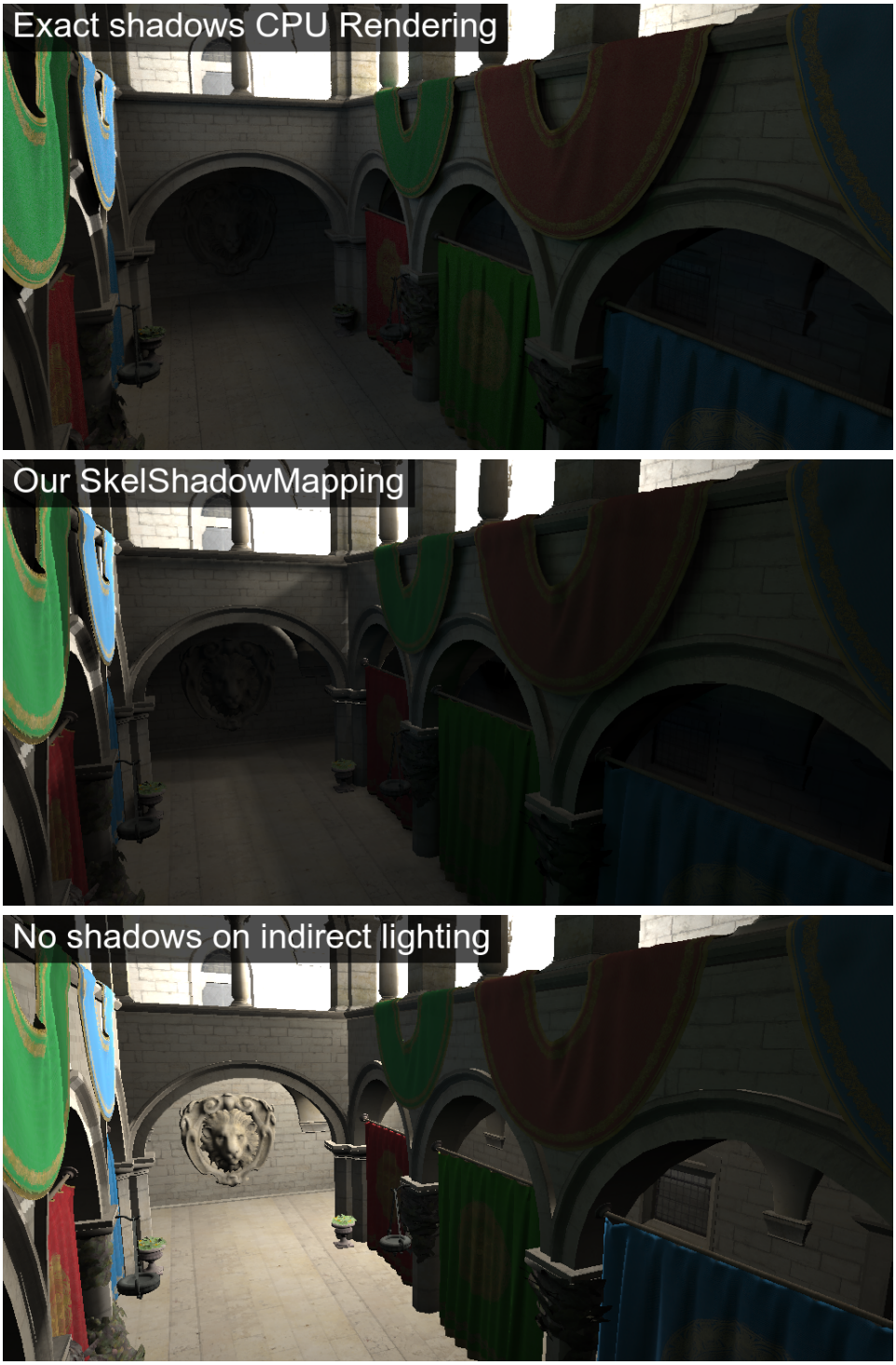


Figure 60 – Configuration Sponza1. Our method overestimates the illumination in that configuration, and produces some light leak near the lion head due to the low resolution of our skeleton node shadow maps. The indirect shadow on the ground is shifted, compared to the exact shadow, and less soft.

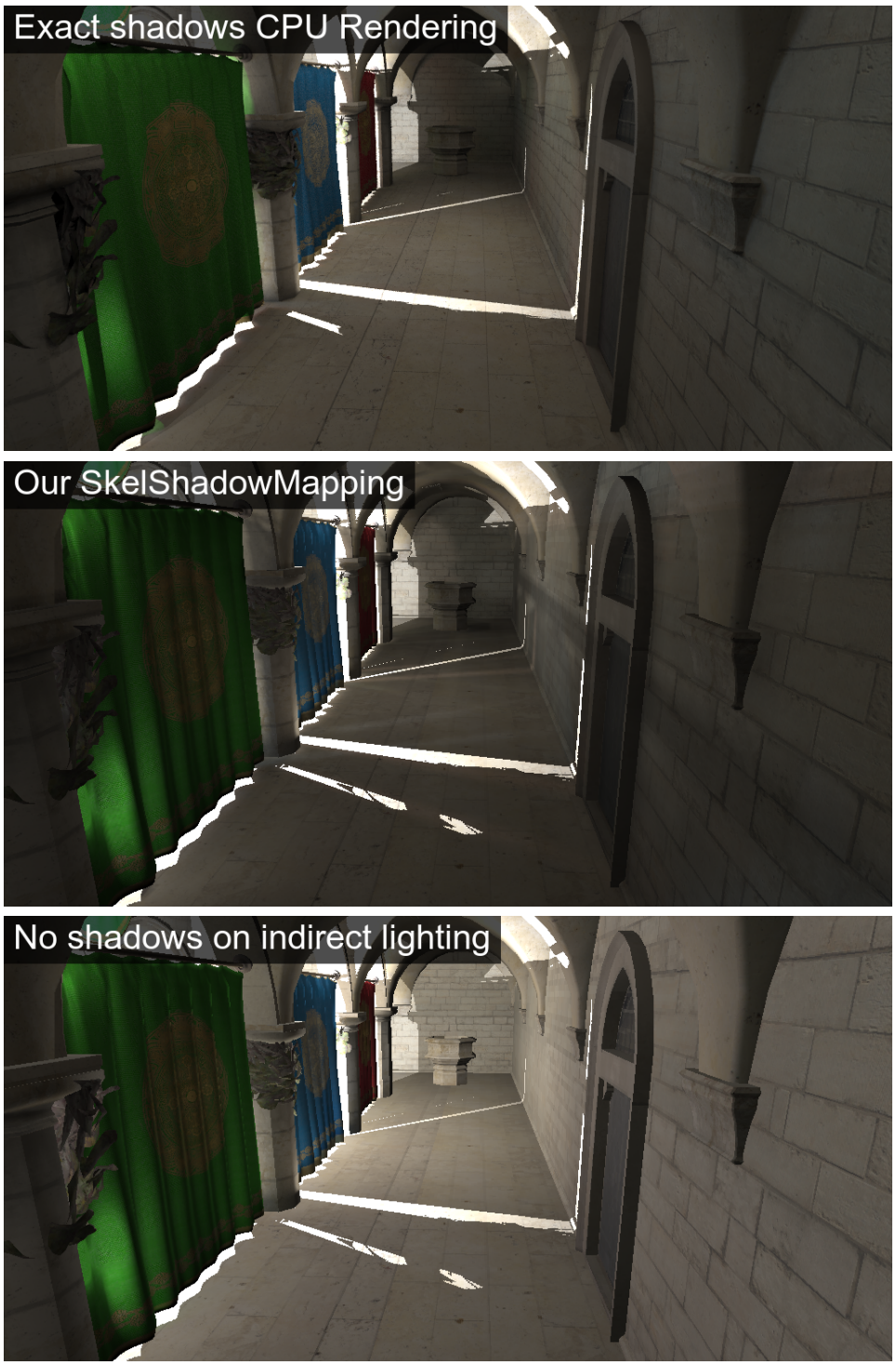


Figure 61 – Configuration Sponza2. A non existent shadow is produced on the right wall by our method.

Figure 62 – Configuration Sponza3. Many non existent light rays are produces on the ground by our method, due to the factorization of visibility for VPLs located on the left corridor.

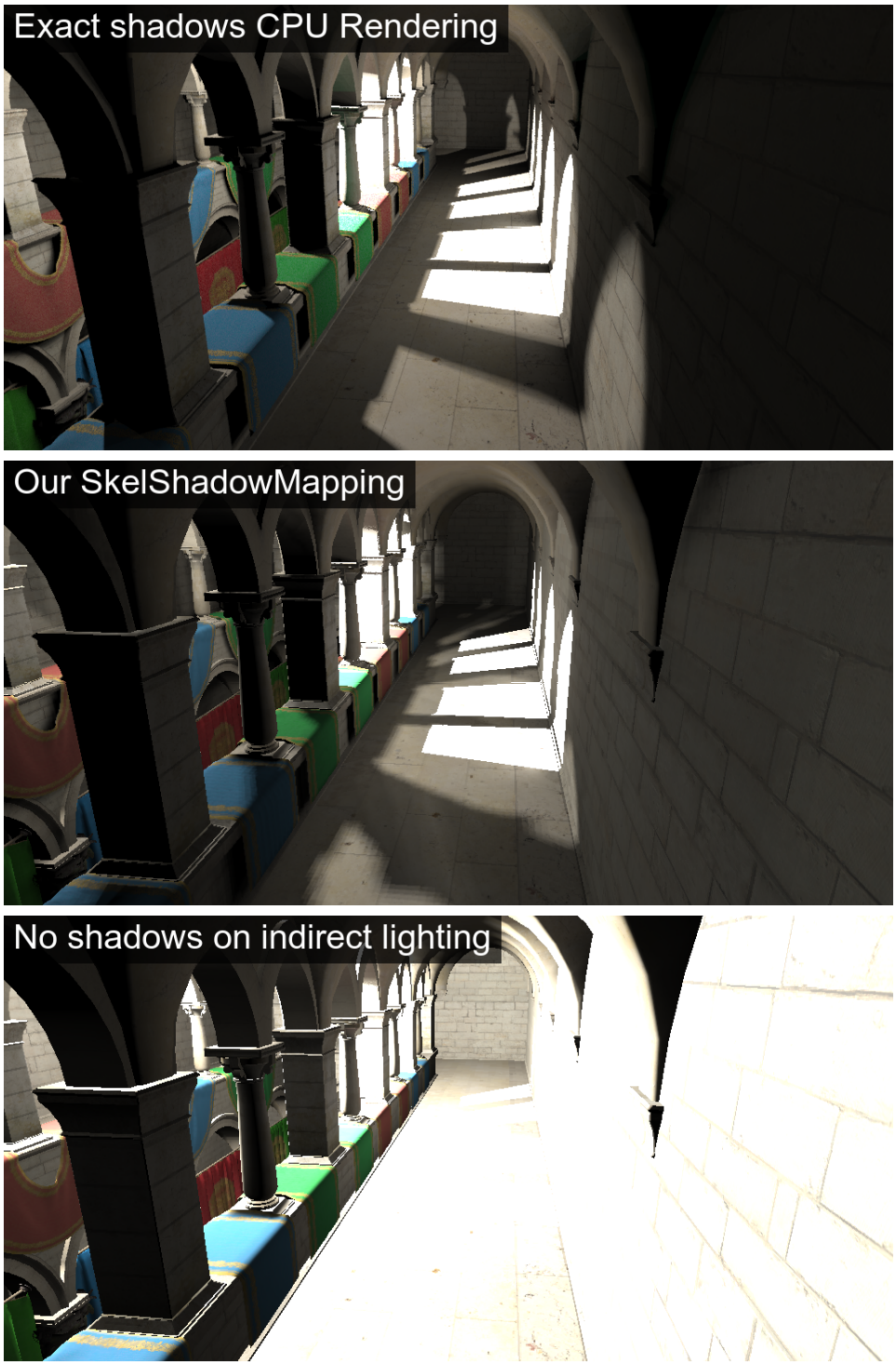


Figure 63 – Configuration Sponza4. While our method is able to produce indirect shadows for pillars, the illumination is overestimated and the shadows are shifted compared to the CPU image. For this viewpoint, it breaks the feeling of realism because the direct illumination part does not align with indirect shadows. Also, the shadows are a little bit aliased because of the low resolution used for our skeleton node shadow maps.

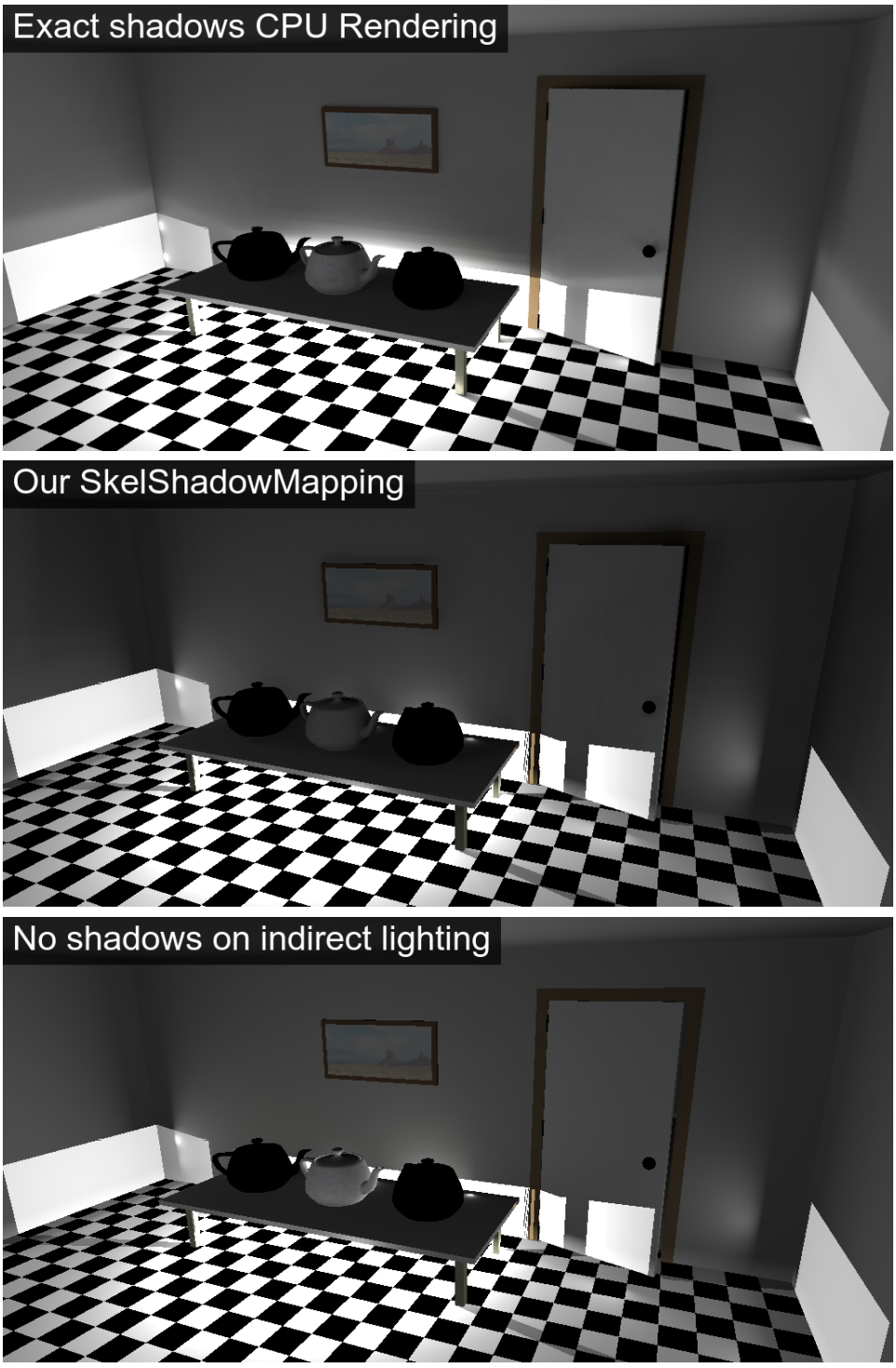


Figure 64 – Configuration Door. In this case, the illumination is underestimated by our method.

Figure 65 – Configuration Apartment1. The illumination is again a little bit overestimated by our method in that configuration.

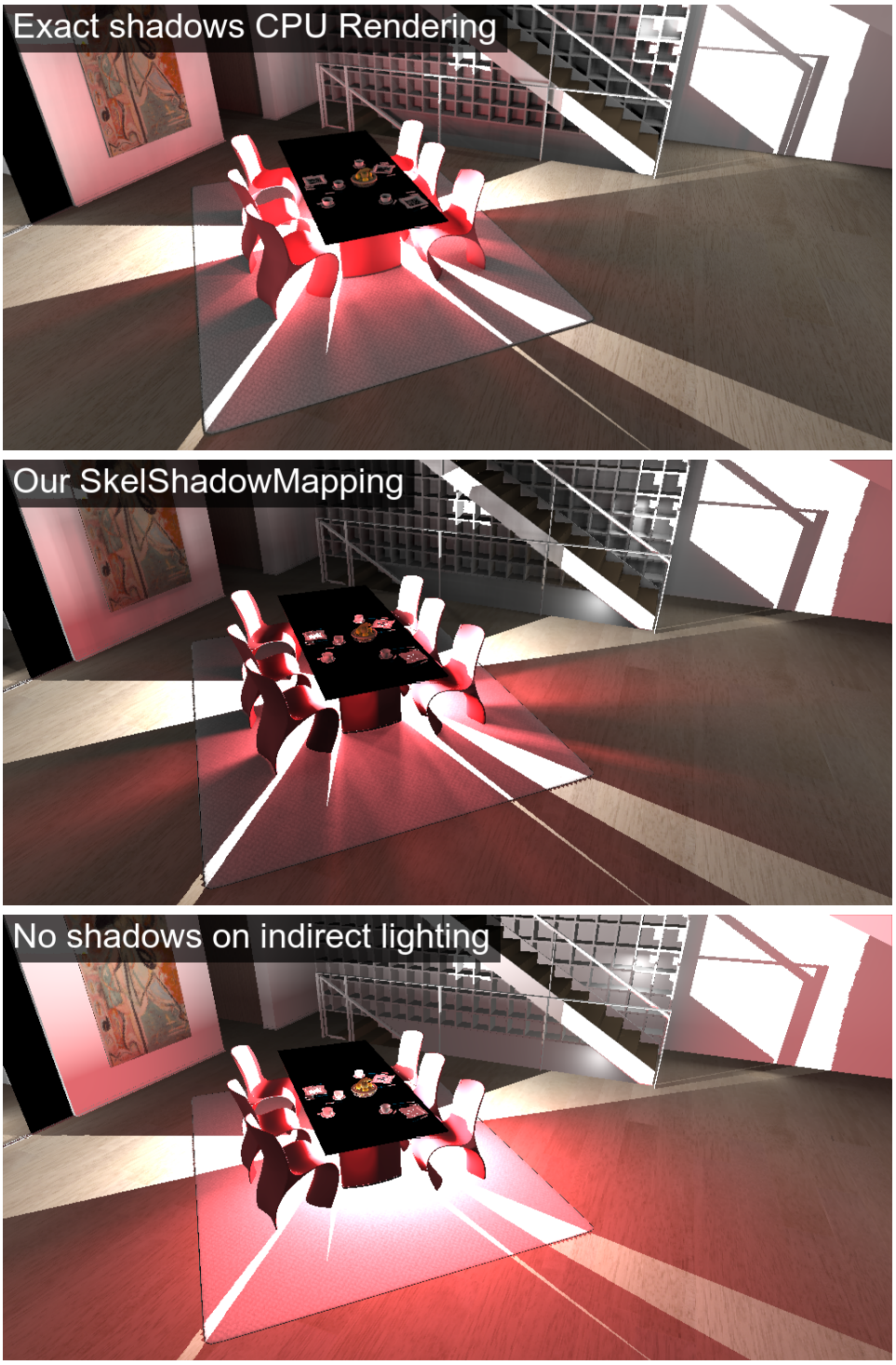


Figure 66 – Configuration Apartment2. This configuration features color bleeding, produced by the reflection of light on the table and chairs. Our method overestimates the bleeding on the ground, but underestimate it on the back wall.

Our method is currently limited to static geometry, as opposed to other more costly solutions for shadow approximations, such as ISMs [RGK+08, REH+11, BBH13] or ManyLODs [HREB11]. A possibility to offset this limitation is to identify node shadow maps that are used at a given frame, and to update them with dynamic objects. A future work is to evaluate the cost and viability of this solution, and to evaluate how it scales with the number of dynamic objects of a given scene.

Finally, we plan to compare our method with the mentioned concurrent ones, in terms of performance and rendering quality.

9

SKELETON BASED RAY IMPORTANCE SAMPLING

This chapter is dedicated to a ray sampling strategy based on topological and geometric information brought by the curvilinear skeleton of the empty space of the scene. The overall goal is to guide ray samples towards important parts of the scene, such as regions containing light sources or the region enclosing the camera. The work exposed in this chapter was initiated by the method presented at the Eurographic 2012 conference [BC12] and partially presented at the DGCI 2013 conference [CNBC13], together with the thinning algorithm of Chapter 6. Section 9.2 presents our sampling strategy to guide rays to a specific region of the scene, where an *importance node* from the skeleton is defined. Extension to multiple importance nodes is detailed in Section 9.3. To obtain an unbiased and robust estimator for Monte Carlo rendering, we explain in Section 9.4 how we combine our sampling strategy with standard BSDF sampling using multiple importance sampling. Section 9.5 explains how we apply our strategy to path tracing and discuss our results. We conclude on this sampling strategy in Section 9.6 and discuss some perspectives for future works.

Contents

9.1 CONTEXT AND GOAL

The sampling strategy presented in this chapter aims at improving importance sampling of rays for local path sampling, detailed in Section 4.1. In the case of highly indirect illumination, resulting from occlusions and strong light sources, local sampling of directions according to the BSDF might not be the best strategy on regions distant from light sources. Indeed, BSDF sampling induces high chances of generating long paths, carrying low contributions or even no contribution due to Russian roulette stopping the sampling. Figure 67 illustrates such difficult situation, where the BSDF sampling strategy gives low probability to directions leading to illuminated parts of the scene.

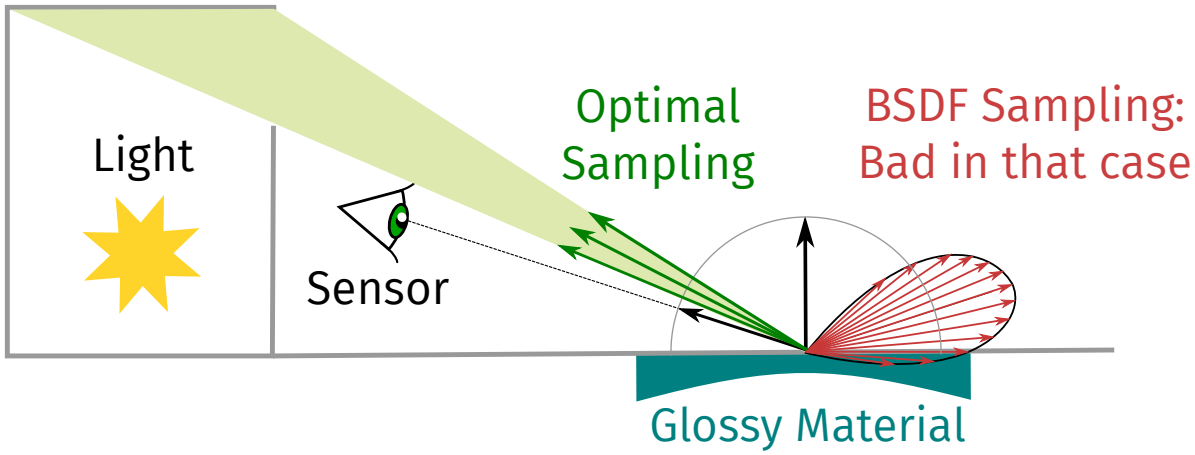


Figure 67 – In that case, sampling directions according to the BSDF contribution gives poor results since few of them would reach the light source.

We propose to take advantage of the skeleton to guide rays towards important parts of the scene, such as light sources for eye sub-path sampling or sensors for light sub-path sampling. Our method is based on a simple assumption: the shorter a path, the higher its contribution. By trying to follow shortest paths along the skeleton graph during ray sampling, we are able to generate short paths connecting the light sources with the camera and then achieve better importance sampling.

9.2 SKELETON BASED RAY SAMPLING STRATEGY

Let **n** be a node from the skeleton representing a source of importance. For example, **n** may be the nearest node from a light source or from the camera. Our goal is to guide ray sampling toward **n**. To achieve that goal, we define the concept of *skeleton importance points*, computed from shortest paths to **n** along the skeleton's graph. These points allow to define a *skeleton importance direction* at each surface point, leading the sampling of rays near **n** in few steps.

9.2.1 *Skeleton importance points*

The first step of our algorithm is to compute the shortest path on the skeleton's graph from each node $\mathbf{n}_k$ to $\mathbf{n}$, using Dijkstra's algorithm [Dij71] for example. The distance between two neighbor nodes $\mathbf{n}_i, \mathbf{n}_j$ from the skeleton graph is simply set to the euclidean distance between their 3D positions.

Let $\mu_k = (\mathbf{n}_k^1 = \mathbf{n}_k, ..., \mathbf{n}_k^n = \mathbf{n})$ denotes the shortest path from a node $\mathbf{n}_k$ to $\mathbf{n}$ and $n' \in [\![1, n]\!]$ the largest index such that $\prod_{j \leq n'} V(\mathbf{n}_k, \mathbf{n}_k^j) = 1$ (i.e. the largest sequence of nodes from μ_k visible from $\mathbf{n}_k$ is $(\mathbf{n}_k^1, ..., \mathbf{n}_k^{n'})$). We define the *skeleton importance point* $I_{\mathbf{n}_k}$ of the node $\mathbf{n}_k$ as the center of mass of the nodes $\mathbf{n}_k^1, ..., \mathbf{n}_k^{n'}$:

$$I_{\mathbf{n}_k} = \frac{1}{n'} \sum_{i=1}^{n'} P_{\mathbf{n}_k^i} \tag{162}$$

where $P_{\mathbf{n}_k^i}$ is the 3D position of the node $\mathbf{n}_k^i$. Figure 68 illustrates the computation of the skeleton importance point of a node.

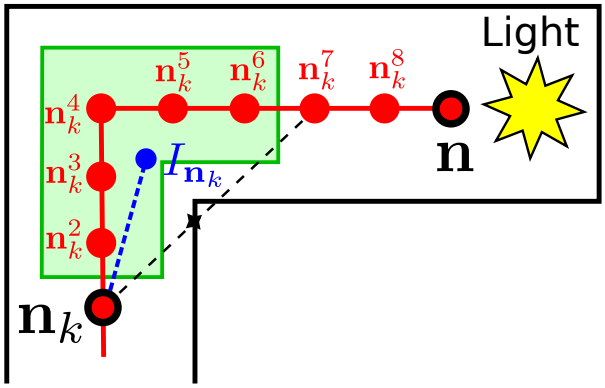


Figure 68 – Illustration of the skeleton importance point $I_{\mathbf{n}_k}$ associated to a node $\mathbf{n}_k$.

9.2.2 *Ray sampling using skeleton importance points*

Let $\mathbf{x} \in \mathcal{M}$ be a surface point of the scene at which we want to sample a direction ω toward $\mathbf{n}$. Let $\mathbf{n}_\mathbf{x}$ be the skeleton's node associated to $\mathbf{x}$, obtained from the node mapping grid (see Section 6.4). The direction $\omega_{\mathbf{x}, I_{\mathbf{n}_\mathbf{x}}}$, pointing from $\mathbf{x}$ to the skeleton importance point $I_{\mathbf{n}_\mathbf{x}}$ is referred as the *skeleton importance direction* of $\mathbf{x}$.

We define the *skeleton sampling pdf* $p_{\text{skel},\mathbf{n},\mathbf{x}}$ associated to $\mathbf{n}$ at $\mathbf{x}$ by:

$$p_{\text{skel},\mathbf{n},\mathbf{x}}(\omega) := \frac{s+1}{2\pi} \max(0, \omega_{\mathbf{x}, I_{\mathbf{n}_\mathbf{x}}} \cdot \omega)^s \tag{163}$$

This pdf defines a cosine lobe centered around the skeleton importance direction. The parameter s is the *skeleton strength*, controlling the focus of the sampling around the importance direction. Figure 69 illustrates the shape of the skeleton sampling pdf.

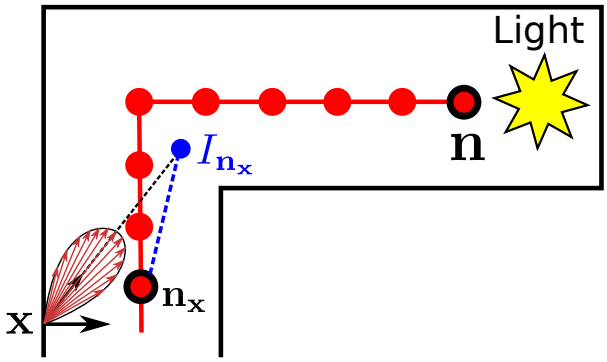


Figure 69 – Illustration of the shape of the sampling strategy $p_{\text{skel},\mathbf{n},\mathbf{x}}$ at a surface point $\mathbf{x}$.

9.3 SAMPLING TOWARD MULTIPLE NODES

Sampling toward multiple importance nodes $\mathbf{n}_1, ..., \mathbf{n}_k$ can be achieved with multiple importance sampling (detailed in Section 3.3.3). These nodes can represent the nearest nodes from each light source, or from hit points obtained by sampling exitant rays from an environment light source for example (see Section 9.5.1 for details about this strategy). We suppose that each importance node $\mathbf{n}_i$ is associated to a importance value $f_i \in \mathbb{R}$, representing its global importance for the sampling. In the case of sampling toward light sources, this importance can be set to the light source power.

Let $\mathbf{x} \in \mathcal{M}$, our goal is to sample a direction ω at $\mathbf{x}$ according to the pdfs $p_{\text{skel},\mathbf{n}_1,\mathbf{x}}, ..., p_{\text{skel},\mathbf{n}_k,\mathbf{x}}$, using MIS. For that, we use the one-sample MIS estimator by defining the selection probability c_i associated to a pdf $p_{\text{skel},\mathbf{n}_i,\mathbf{x}}$ as:

$$c'_i := \frac{f_i}{d_{\text{skel}}(\mathbf{n_x}, \mathbf{n}_i)^2} \tag{164}$$

$$c_i := \frac{c'_i}{\sum_{j=1}^{k} c'_j} \tag{165}$$

where $d_{\text{skel}}(\mathbf{n_x}, \mathbf{n}_i)$ is the shortest distance along the skeleton's graph from the node $\mathbf{n_x}$, associated to the surface point $\mathbf{x}$, to the importance $\mathbf{n}_i$. This distance is computed at the same time as the shortest path computation, described in the previous section. These selection probabilities mimics the attenuation of power according to the inverse-square law.

We use the balance heuristic to define MIS weights:

$$w_i(\omega) := \frac{c_i . p_{\text{skel},\mathbf{n}_i,\mathbf{x}}(\omega)}{\sum_{j=1}^{k} c_j . p_{\text{skel},\mathbf{n}_j,\mathbf{x}}(\omega)} \tag{166}$$

The final sampling strategy, combining each importance node individual strategy, is referred as p_{skel} and we have:

$$p_{\text{skel}}(\omega) = \sum_{j=1}^{k} c_j . p_{\text{skel},\mathbf{n}_j,\mathbf{x}}(\omega) \tag{167}$$

9.4 COMBINATION WITH BSDF SAMPLING

The sampling strategy p_{skel} is biased by construction since it cannot sample all contributing directions at a given point. Even if it could, ignoring importance sampling of the BSDF can lead to high variance in the estimation at shiny materials. To overcome these issues, we add another step of MIS, combining our skeleton sampling strategy p_{skel} with the BSDF sampling strategy $p_{\sigma^\perp,f_s}$ introduced in Section 4.1.2. We introduce a parameter $c_{\text{skel}} \in [0,1]$ to control the probability of using the skeleton sampling strategy p_{skel}, thus $c_{\text{bsdf}} = 1 - c_{\text{skel}}$ is the probability of using the BSDF sampling strategy $p_{\sigma^\perp,f_s}$.

With the balance heuristic, the MIS weights are expressed as:

$$w_{\text{skel}}(\omega) := \frac{c_{\text{skel}} . p_{\text{skel}^\perp}(\omega)}{c_{\text{skel}} . p_{\text{skel}^\perp}(\omega) + c_{\text{bsdf}} . p_{\sigma^\perp,f_s}(\omega)} \tag{168}$$

$$w_{\text{bsdf}}(\omega) := \frac{c_{\text{bsdf}} . p_{\sigma^\perp,f_s}(\omega)}{c_{\text{skel}} . p_{\text{skel}^\perp}(\omega) + c_{\text{bsdf}} . p_{\sigma^\perp,f_s}(\omega)} \tag{169}$$

where $p_{\text{skel}^\perp}(\omega)$ is the probability density with respect to projected solid angle associated to ω by the skeleton sampling strategy, expressed by (see Section 3.1.3):

$$p_{\text{skel}^\perp}(\omega) = \frac{p_{\text{skel}}(\omega)}{|\omega \cdot n_{\mathbf{x}}|} \tag{170}$$

9.5 APPLICATION TO PATH TRACING

This section presents an extended version of the original algorithm we proposed in [CNBC13], with the goal of improving its robustness and its generality regarding the types of light sources used to illuminate the scene. While the original method was limited to positional light sources (area lights and point lights), this new version is also able to deal with directional and environment light sources.

9.5.1 *Computation of importance nodes*

To apply our sampling strategy to path tracing, we just have to define a set of importance nodes with their importance values and then proceed as detailed in previous sections. Our goal is to sample rays towards bright parts of the scene, i.e directly illuminated, since this is where next event estimation creates paths with non-null contribution (as explained in Section 4.2.3). To achieve that goal, we associate to each node $\mathbf{n}$ an importance value that estimates the incident emitted power on the set of surface points $\mathcal{M}_{\mathbf{n}}$ that are mapped to the node. Each node for which this estimation results in a non-null value is used as an importance node.

The set of points $\mathcal{M}_{\mathbf{n}}$ is not accessible directly: we can know if a point belongs to $\mathcal{M}_{\mathbf{n}}$, using the node mapping grid, but not sample efficiently points from that set. Consequently, we

derive an expression of this importance value such that it can be estimated with Monte Carlo integration by sampling rays from light sources.

Let $\mathcal{M}_{\mathbf{n}}$ denotes the set of surface points that are mapped to $\mathbf{n}$ by the node mapping grid:

$$\mathcal{M}_{\mathbf{n}} \coloneqq \{\mathbf{x} \in \mathcal{M} \mid \mathbf{n}_{\mathbf{x}} = \mathbf{n}\} \tag{171}$$

The incident emitted power on that set is expressed by:

$$\Phi_{\mathrm{e,i}}(\mathcal{M}_{\mathbf{n}}) \coloneqq \int_{\mathbf{x}\in\mathcal{M}_{\mathbf{n}}} \int_{\omega_{\mathrm{i}}\in\mathcal{S}^2} L_{\mathrm{e,i}}(\mathbf{x},\omega_{\mathrm{i}}) d\mathcal{A}(\mathbf{x}) d\sigma^{\perp}_{\mathbf{x}}(\omega_{\mathrm{i}}) \tag{172}$$

where $L_{\mathrm{e,i}}$ is the incident emitted radiance, defined as $L_{\mathrm{e,i}}(\mathbf{x},\omega_{\mathrm{i}}) \coloneqq L_{\mathrm{e}}(\mathbf{x}_{\mathcal{M}}(\omega_{\mathrm{i}}), -\omega_{\mathrm{i}})$.

Equation (172) can be rewritten with respect to area measure, by integrating over surfaces of light sources $\mathcal{M}_L$ only:

$$\Phi_{\mathrm{e,i}}(\mathcal{M}_{\mathbf{n}}) = \int_{\mathbf{x}\in\mathcal{M}_{\mathbf{n}}} \int_{\mathbf{y}\in\mathcal{M}_L} L_{\mathrm{e}}(\mathbf{y},\mathbf{x}) G(\mathbf{x},\mathbf{y}) d\mathcal{A}(\mathbf{x}) d\mathcal{A}(\mathbf{y}) \tag{173}$$

We now rewrite the equation by replacing the outer integral, expressed over $\mathcal{M}_{\mathbf{n}}$, by an integral expressed over the whole scene $\mathcal{M}$ with the indicator function $\mathbb{1}_{\mathcal{M}_{\mathbf{n}}}$ as an extra factor in order to force the integrand to be null outside $\mathcal{M}_{\mathbf{n}}$:

$$\Phi_{\mathrm{e,i}}(\mathcal{M}_{\mathbf{n}}) = \int_{\mathbf{x}\in\mathcal{M}} \int_{\mathbf{y}\in\mathcal{M}_L} L_{\mathrm{e}}(\mathbf{y},\mathbf{x}) G(\mathbf{x},\mathbf{y}) \mathbb{1}_{\mathcal{M}_{\mathbf{n}}}(\mathbf{x}) d\mathcal{A}(\mathbf{x}) d\mathcal{A}(\mathbf{y}) \tag{174}$$

Finally, we apply another change of measure for the outer integral, from surfaces to projected solid angle at light sources:

$$\Phi_{\mathrm{e,i}}(\mathcal{M}_{\mathbf{n}}) = \int_{\mathbf{y}\in\mathcal{M}_L} \int_{\omega_{\mathrm{o}}\in\mathcal{S}^2} L_{\mathrm{e}}(\mathbf{y},\omega_{\mathrm{o}}) \mathbb{1}_{\mathcal{M}_{\mathbf{n}}}(\mathbf{y}_{\mathcal{M}}(\omega_{\mathrm{o}})) d\sigma^{\perp}_{\mathbf{y}}(\omega_{\mathrm{o}}) d\mathcal{A}(\mathbf{y}) \tag{175}$$

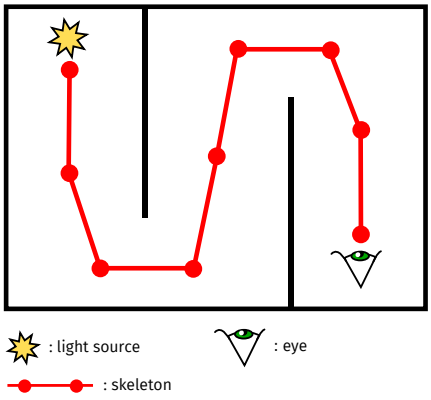
This value can be estimated by tracing N random rays $((\mathbf{y}_i,\omega_i))_{i\in[\![1,N]\!]}$, leaving lights sources and distributed according to the joint pdf $p \coloneqq p_{L_{\mathrm{e}}}.p_{\sigma^{\perp},L_{\mathrm{e}}}$, with Monte Carlo integration:

$$\widehat{\Phi_{\mathrm{e,i}}}(\mathcal{M}_{\mathbf{n}}) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \frac{L_{\mathrm{e}}(\mathbf{y}_i,\omega_i)\mathbb{1}_{\mathcal{M}_{\mathbf{n}}}(\mathbf{y}_{i\mathcal{M}}(\omega_i))}{p(\mathbf{y}_i,\omega_i)} \tag{176}$$

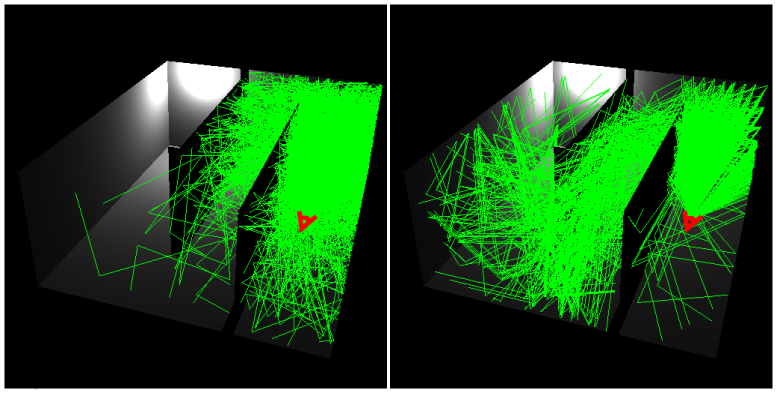
Since skeleton mapping defines a partitioning of the surfaces of the scene, this value can be computed for all nodes at the same time. $\widehat{\Phi_{\mathrm{e,i}}}(\mathcal{M}_{\mathbf{n}})$ is initialized to zero for all nodes. Then for each sampled ray $(\mathbf{y}_i,\omega_i)$, the contribution $\frac{L_{\mathrm{e}}(\mathbf{y}_i,\omega_i)}{p(\mathbf{y}_i,\omega_i)}$ is added to the value $\widehat{\Phi_{\mathrm{e,i}}}(\mathcal{M}_{\mathbf{n}})$ for which $\mathbf{n} = \mathbf{n}_{\mathbf{y}_{i\mathcal{M}}(\omega_i)}$.

Each node $\mathbf{n}$ such that $\widehat{\Phi_{\mathrm{e,i}}}(\mathcal{M}_{\mathbf{n}}) \neq 0$ is then used as an importance node to drive skeleton importance sampling, with the importance value $f \coloneqq \widehat{\Phi_{\mathrm{e,i}}}(\mathcal{M}_{\mathbf{n}})$.

Note that even if Equation (175) is expressed over light source surfaces (for derivation of the formula only), the formula is general and can be also used for non-physically based light sources, such as point lights, directional lights and environment lights, by extending the domain of integration $\mathcal{M}_L$.



(a) Corridor layout



(b) left: BSDF sampling, right: skeleton sampling

Figure 70 – Corridor scene.

9.5.2 *Results and discussion*

We compare our sampling strategy against BSDF sampling, applied to the path tracing algorithm. We demonstrate results for various values of the skeleton strength parameter, and discuss in the conclusion some perspectives regarding its automatic computation. All results were computed with two CPUs Intel Xeon E5-2650 hexa-core at 2.0 Ghz. Reference images have been rendered with bidirectional path tracing, for various rendering times depending on the configuration. For our most complicated settings, reference images still present some noise, despite several days of computation.

For each comparison, the number of rendering iterations is indicated in parenthesis in the image, as well as the mean absolute error with the reference. Convergence curves are shown in Figure 77 and gives a lower error for our method for all presented configurations.

CORRIDOR *Corridor* is a diffuse scene that we designed to illustrate, in a simple setting, the difficulty of reaching a distant light source with BSDF sampling. Figure 70a illustrates the scene layout and Figure 70b compares the ray distribution generated by BSDF sampling and by our skeleton sampling strategy. While BSDF sampling have difficulties to connect the eye with the light source, our sampling easily guide rays in the good direction. Figure 71 shows a result produced by our algorithm after 20 minutes rendering in which we observe a good noise reduction, compared to standard path tracing. An important observation is the low number of rendering iterations (4513) for our method, compared to path tracing (10694). This is due to the cost of sampling a power cosine distribution, which increases for high values of the exponent. Rendering the scene with a skeleton strength set to 1 allows to compute 8307 iterations but also present more noise in the result.

SPONZA PLUGING *Sponza pluging* is a diffuse configuration for the famous Sponza scene, remodeled by Crytek. The scene is illuminated by two small area light, located far from the view point and directly illuminating a small part of the surface, making it difficult for standard path tracing. For this setting, our skeleton sampling strategy slightly improve

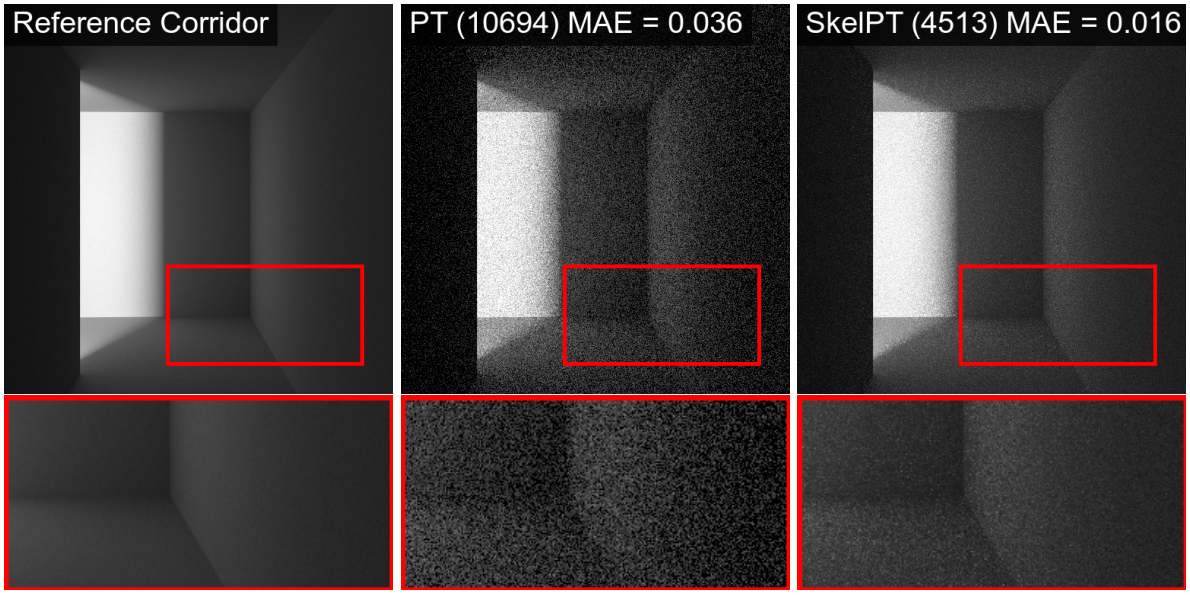


Figure 71 – Corridor scene rendering, after 20 min rendering. Skeleton strength set to 16 for skelPT.

the error compared to BSDF sampling, but the noise reduction is visible in the result image (Figure 72). The skeleton strength has been set to 8. Overall, our experiments tend to indicate that a high skeleton strength is required for diffuse materials, when the source of importance is located far from the point at which a ray is sampled.

SPONZA GLOSSY For this configuration, we increased the glossyness of the ground of Sponza. The light source has a high power and is located in the opposite corridor. The view point presented in Figure 73 is quite difficult to render efficiently using only BSDF sampling since the glossiness of the ground lead the sampling of rays in the wrong global direction: toward the end of the corridor. Our strategy allows to sample rays towards the openings that lead to bright parts of the scene. For glossy materials, the skeleton strength has to be lower than for diffuse ones, in order to slightly change the orientation of sampled rays without completely ignoring the glossy component.

PLANTS This scene contains many occlusions and small light sources, modeled as light bulbs. While next event estimation is mandatory in this kind of configuration, it is not enough to sample many contributing paths and BSDF sampling results in a dark image after one hour rendering, compared to the reference (Figure 74). Our strategy produces an image closer to the reference in terms of illumination, by successfully sampling paths leaving narrow parts of the empty space, such as regions under tables or chairs.

APARTMENT This scene is lit by a large but distant area light source, located two floors above from the view point. The ground is glossy and standard path tracing renders it almost completely black after one hour. While our sampling strategy also produces a noisy result, the rendering of the ground offers a closer match to the reference (Figure 75).

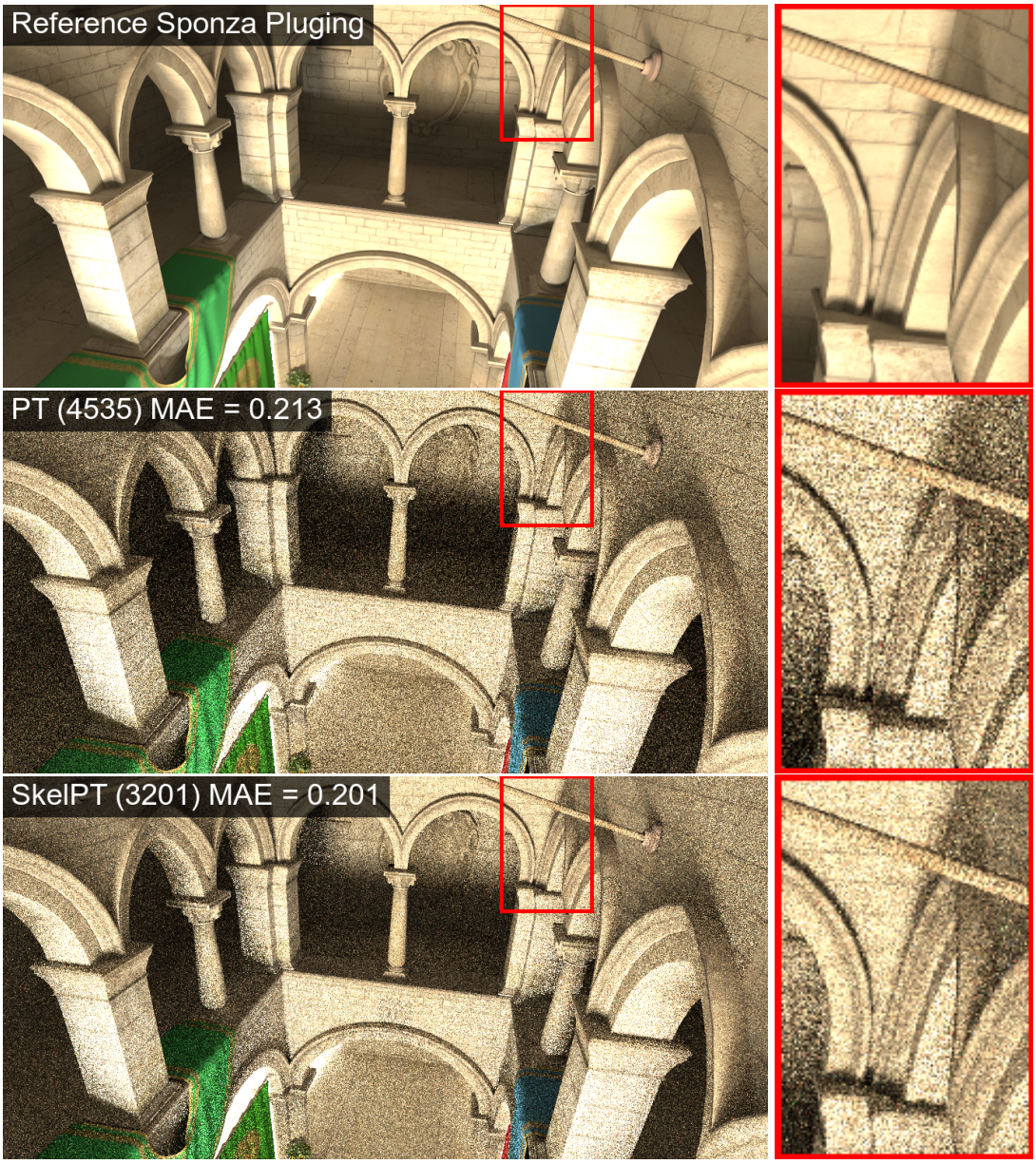


Figure 72 – 30 min comparison.

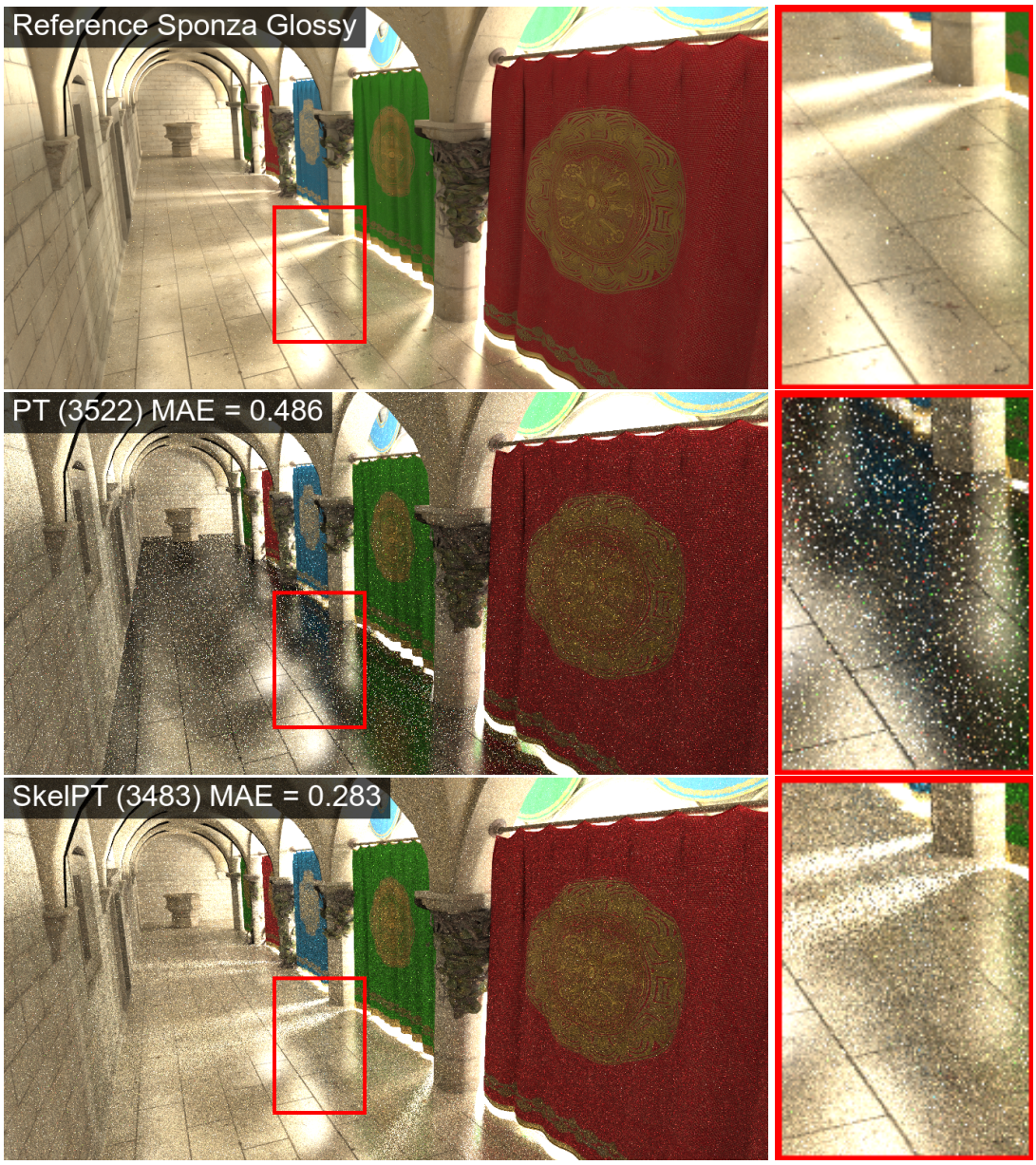


Figure 73 – 30 min comparison

Figure 74 – 1 hour comparison

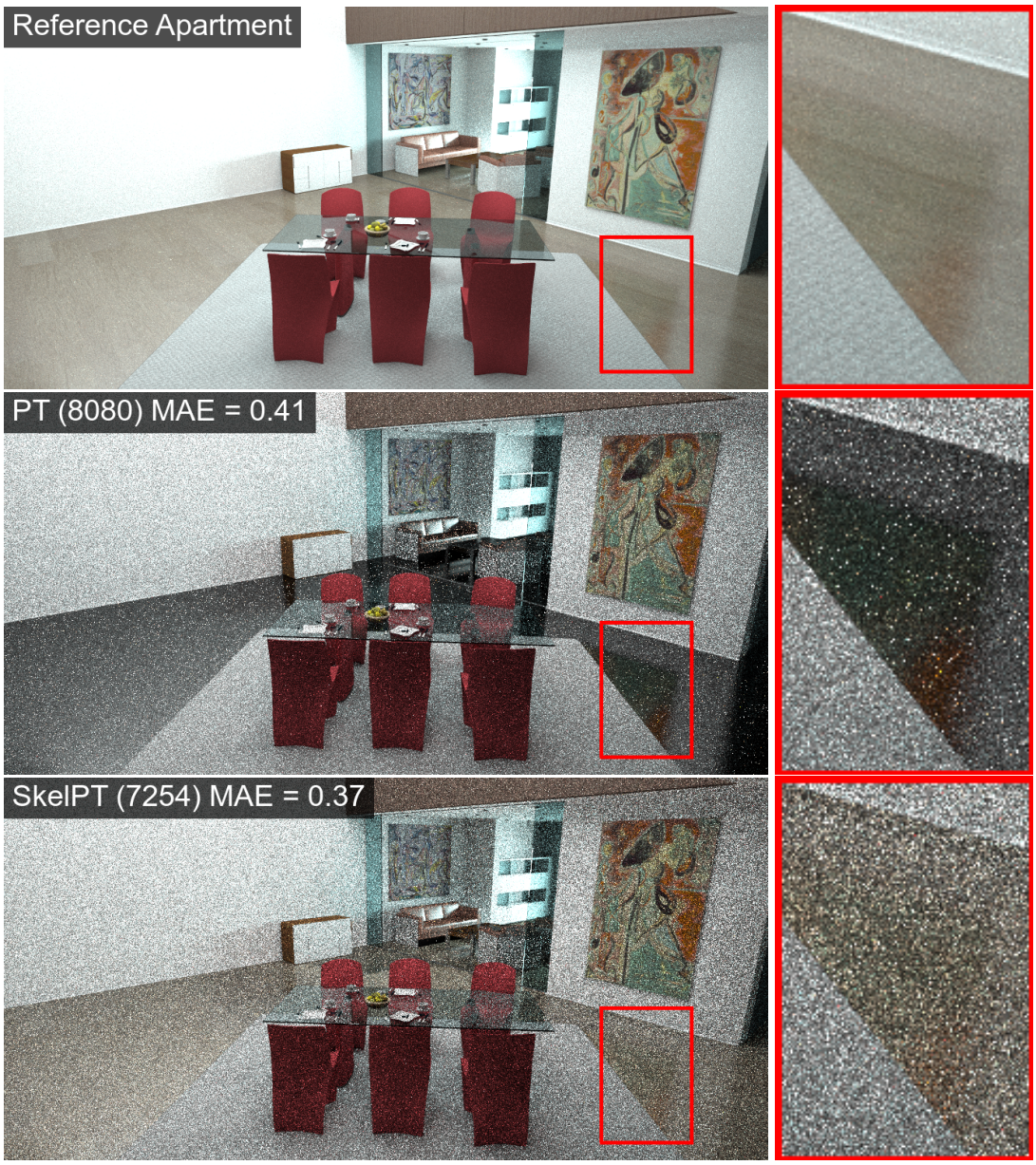


Figure 75 – 1 hour comparison

Figure 76 – 15 min comparison

SPONZA DARK On this configuration for Sponza, we illustrate three results computed with our method for different values of the skeleton strength parameter (Figure 76). As we increase the value of the parameter, the noise in the blue frame is reduced but the noise in the red frame is increased. These results clearly demonstrate that, to take the best from the skeleton, the value of the skeleton strength parameter cannot be set globally and should be automatically computed from prior information about the scene geometry, material and illumination at each surface point.

9.6 CONCLUSION AND PERSPECTIVES

We proposed a new ray sampling strategy based on topological and geometric information through the use of our curvilinear skeleton of the empty space of the scene. This strategy is simple to implement, efficient and provides an alternative and original solution to address the problem of importance ray sampling. As opposed to previous approaches [Jen95, PPI98, HP02, SL06, VKŠ+14] described in Section 4.1.4, our method is not based on particle density estimation, but purely on information extracted from the empty space of the scene. A comparison of our technique against these concurrent methods is planned for future works, as well as the adaptation of the strategy to other path sampling rendering algorithms presented in Chapter 4.

Our skeleton based strategy is particularly good for glossy materials, when the BSDF sampling strategy is focused around directions that lead to dark parts of the scene instead of bright ones. Overall, our results demonstrate good improvements over BSDF sampling, but the rendering quality is highly dependent on the skeleton strength parameter. As discussed in the result section, setting globally this parameter does not provide an optimal use of the sampling strategy. For example, on the apartment scene, the skeleton strength should be set

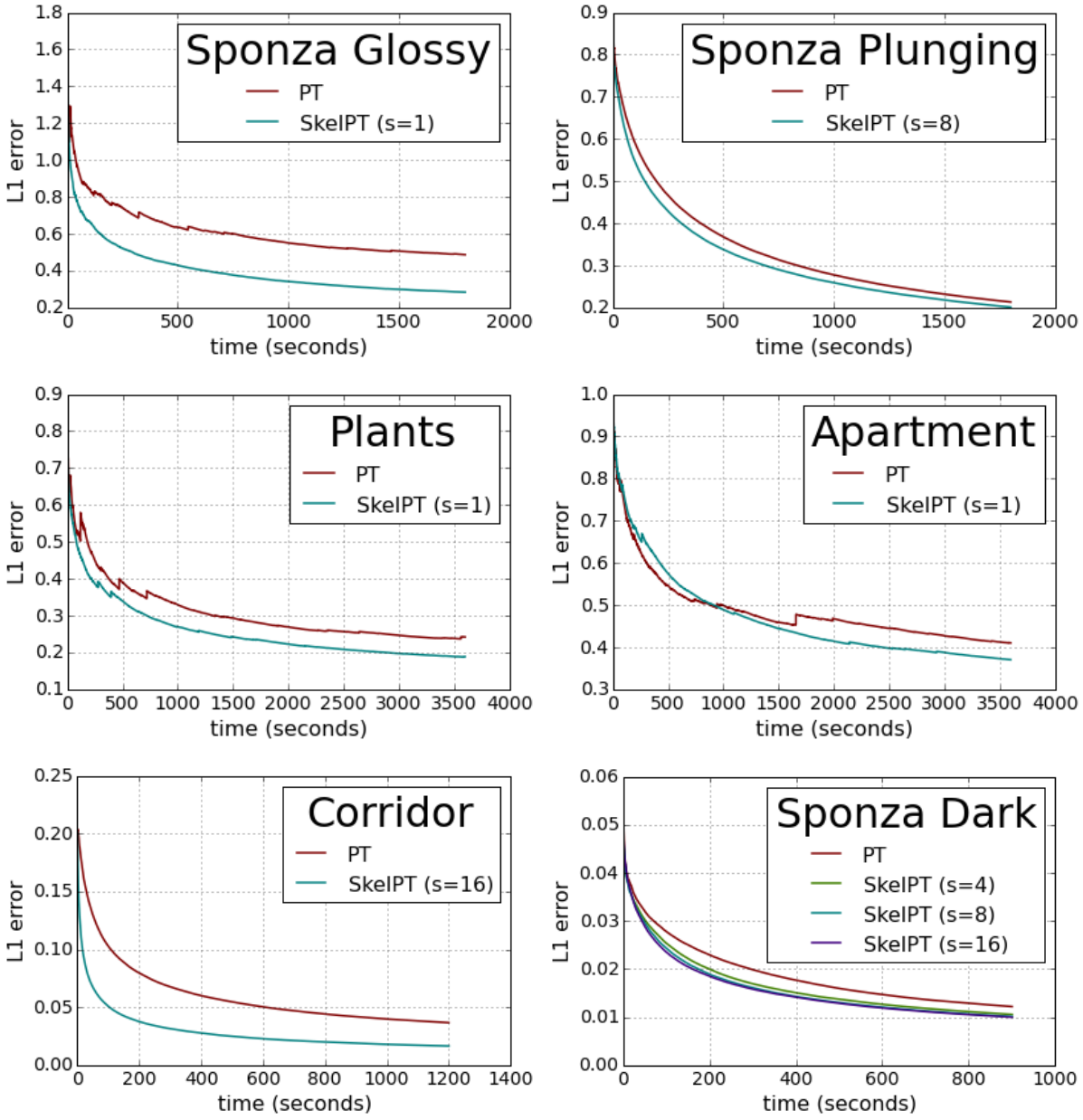


Figure 77 – Convergence curves for the presented results. The value of the skeleton strength parameter s is indicated for each configuration.

higher at diffuse materials in order to get a better noise reduction at these points. Optimally, this parameter should also vary with the position in the scene: higher far from importance nodes and lower near them. We plan on future works to further investigate optimal values for this parameter and its relation to the local geometric configuration, in order to provide an automatic computation of its value and therefore a parameter-free sampling strategy. Finally, we experimented energy diffusion heuristics along the skeleton graph [NCB12], in order to replace the shortest paths solution with a more robust one for the identification of main light streams, and we plan to extend further these research to apply them to the works presented in this chapter.

10

PORTAL EXTRACTION BASED ON AN OPENING LABELING FOR RAY SAMPLING

This chapter addresses the same problem as Chapter 9: ray sampling toward important regions of the scene in presence of complex occlusions. Instead of a curvilinear skeleton, we take advantage of an opening map [Vin94, Coe12] of the empty space to extract useful information for driving ray sampling. An opening map measures at each voxel the local thickness of the empty space and can be used to extract portals separating regions that are usually hard to connect with local ray sampling. Note that we already use an opening map to compute the skeleton of the scene (as explained in Section 6.2.2) since it can also be exploited to preserve the shape of the empty space in the skeleton. The portal extraction method detailed in this chapter was presented at the ISMM 2015 conference [NB15a].

Section 10.1 explains more in depth the motivation for using an opening map to extract portals of the empty space. To compute these portals, we propose in Section 10.3 to construct a labeling of the digital empty space according to the opening map, such that two neighboring regions are characterized by a high difference in their thickness. Our first rendering experiments, applied to path tracing, are presented in Section 10.4. While our first results are promising, discontinuity problems between neighbor sampling strategies at surfaces of the scene arise in this application. We discuss ideas to improve the labeling and the mixture between sampling strategies that will drive our future works to solve these issues. We conclude this chapter in Section 10.5 and discuss some perspectives.

Contents

10.1 MOTIVATION

We begin by explaining why the opening map can be interesting for ray sampling, through a discussion over a scene commonly used in computer graphics.

The Ajar Door scene (Figure 78) is composed of two large rooms separated by a door slightly opened. For a configuration where a light source is located in one room and the camera in the other room, the illumination becomes hard to sample efficiently. Indeed, blindly tracing random paths starting at the camera will end up with many paths not reaching the room containing the light source.

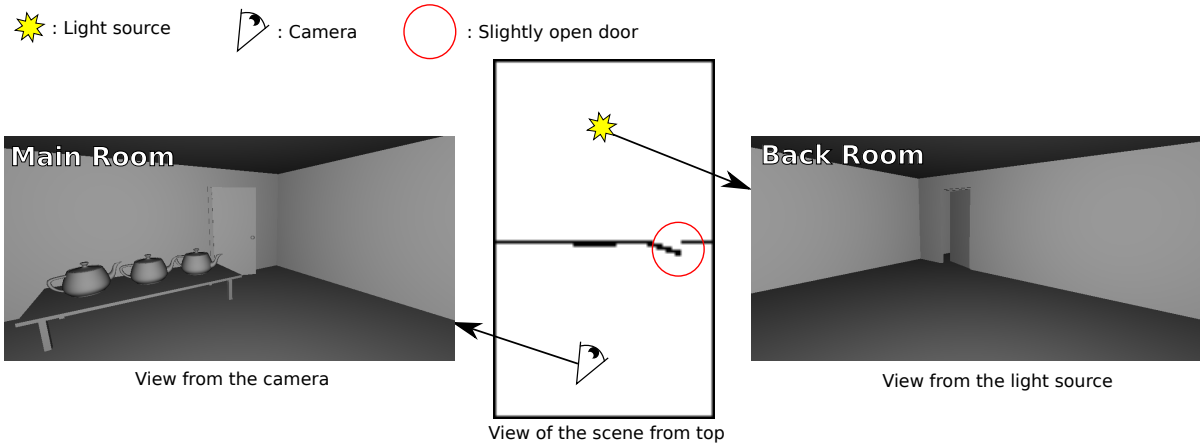


Figure 78 – Ajar Door scene. The center image is a cut along the vertical axis of a voxelization of the scene (black pixels are from surfaces and white pixels are from the empty space). The red circle encloses the small aperture that rays must traverse to go from one room to the other.

The opening map of the digital empty space E_D tells us for each voxel $\mathbf{x}$ the radius of the maximal ball inscribed in E_D that contains $\mathbf{x}$ (Figure 79). We observe that the aperture highlighted in red is characterized by lower values for this opening map than the two large rooms.

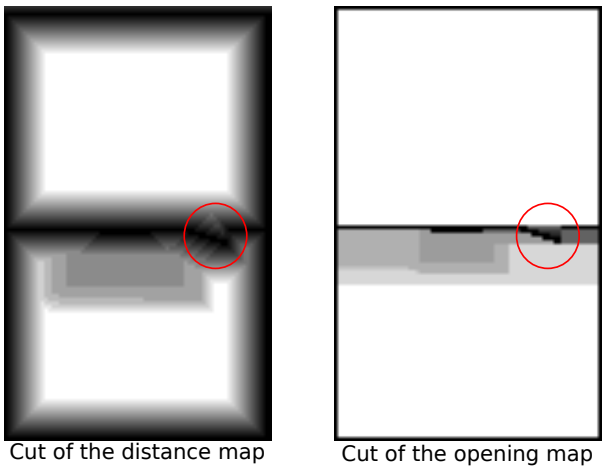


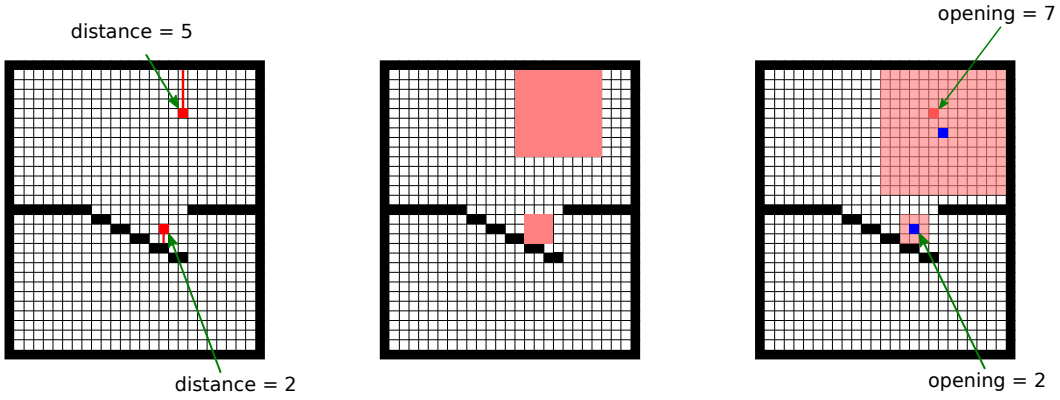
Figure 79 – By dilating the distance map (left), we obtain the opening map (right) of the empty space. This map enhances narrow regions which are difficult to explore by tracing random rays.

From this observation, we present a new labeling method that partitions the empty space in regions based on the opening map. More importantly we extract *opening portals* that are surfaces separating a region from its neighbor regions. Sampling these portals offers the possibility of tracing rays immediately leaving a region, and thus to efficiently explore the empty space, searching for important regions regarding light transport simulation.

10.2 RELATED WORKS

The extraction of portals separating regions of a 3D scene is a difficult problem that has been addressed by several authors over the past years. Cohen-Or et al. [COCSD03] provide a rich overview of various methods to speed-up visibility computations on highly occluded scenes. In particular, rendering algorithms driven by *cells-and-portals* acceleration structures were initiated by the works of Airey et al. [ARB90] and Teller et al. [TS91, Tel92, TFFH94]. The goal of these works is to split the scene into 3D cells, separated by portals, in order to efficiently propagate illumination from cell to cell. This strategy allows to handle large 3D models that cannot fit in main memory since the computation can be performed iteratively and involve, at each step, only few cells. However, these methods are mainly focused on large architectural buildings that are composed of axis aligned polygons. Indeed, these works rely on a binary space partitioning scheme to separate the scene in cells, which is known to have a high complexity for non axial scenes. To address this issue, Meneveaux et al. [MMB98] proposed a model-based partitioning solution that can be applied to non axial buildings, but limited to walls perpendicular to the ground. By applying geometric rules driven by the model, their method is able to construct cells that make sense regarding the topology of the input scene, such as the set of rooms and corridors of a building. Together with the partitioning strategy, they propose a solution to extract portals that separate cells. A cells-and-portals structure has been used by Fradin et al. [FMH05] to efficiently compute photon mapping on buildings composed of billions of polygon, where individual cells are loaded from disk and processed iteratively. In their case, the cells-and-portals structure is optimal, but constructed manually using a dedicated topological modeller, which is a tedious and time consuming task. More recently, Maria et al. [MHA14] proposed a fast ray tracing approach for cells-and-portals models, which take advantage of all topological properties of the model to perform a very efficient traversal. They also exploit the structure to compute potentially visible lights for each cell in order to speed up direct illumination computation.

Despite good performance, these approaches are mainly limited to architectural scenes, composed of vertical planar walls, which is not the case for general 3D scenes (even buildings can be composed of non vertical architectural elements, such as an hemispherical roof). As opposed to these methods, our strategy makes no assumption on the scene's geometry. Partitioning of the scene and portals extraction are entirely computed on a digital representation of the empty space of the input scene. This representation allows to obtain portals as sets of 2D faces of voxels, and thus to avoid dealing with the complexity of arbitrary polygons constituting the scene.



(a) 26-distance map (b) 26-maximal balls (c) 26-opening

Figure 80 – Illustration of the discrete quantities defined in Section 10.3.1.

10.3 OPENING BASED LABELING

We first present the mathematical background related to the definition of opening maps according to the 26-distance. Then we study a simple labeling obtained from connected flat regions of the opening map and demonstrate that too many false positive regions are produced regarding our problem of ray sampling. Consequently, we define a new labeling based on a deeper analysis of the structure of the opening map and the way it is computed by a standard algorithm. Our results illustrates that this labeling allows to extract meaningful portals for ray sampling, defined as two dimensional surfaces separating regions of the labeling.

10.3.1 *Mathematical background*

First recall the definitions, introduced in Section 6.1.2, related to the digital empty space E_{D}:

- The set of voxels composing the digital scene is referred as $\mathcal{M}_{\mathrm{D}} \subset \mathbb{Z}^3$.
- The digital scene is included in the digital grid $G_{w,h,d} := [\![1, w]\!] \times [\![1, h]\!] \times [\![1, d]\!] \subset \mathbb{Z}^3$ of resolution (w, h, d), that depends on the voxelization resolution.
- The digital empty space is defined as $E_{\mathrm{D}} := G_{w,h,d} \setminus \mathcal{M}_{\mathrm{D}}$.

We now define quantities related to the *26-opening map*. Figure 80 provides an illustration of these quantities, in 2D for clarity.

Definition 10.1. If $\mathbf{x} = (x_1, x_2, x_3)$, $\mathbf{y} = (y_1, y_2, y_3) \in \mathbb{Z}^3$ are two voxels, we denote by $d_\infty(\mathbf{x}, \mathbf{y}) := \max(|x_1 - y_1|, |x_2 - y_2|, |x_3 - y_3|)$ the *26-distance* (also known as *chessboard distance* or *Chebyshev distance*) between $\mathbf{x}$ and $\mathbf{y}$.

Definition 10.2. The 26-distance map to $\mathcal{M}_D$, $\mathcal{D}_\infty : G_{w,h,d} \to \mathbb{Z}^+$, is the function defined by:

$$\mathcal{D}_\infty(\mathbf{x}) := \min_{\mathbf{y} \in \mathcal{M}_D} d_\infty(\mathbf{x}, \mathbf{y}) \tag{177}$$

Note that we have $\mathbf{x} \in \mathcal{M}_D \Leftrightarrow \mathcal{D}_\infty(\mathbf{x}) = 0$.

Definition 10.3. The 26-maximal ball $\mathcal{B}_\infty(\mathbf{x})$ centered in $\mathbf{x}$ is the set:

$$\mathcal{B}_\infty(\mathbf{x}) := \{\mathbf{y} \in G_{w,h,d} \mid d_\infty(\mathbf{x}, \mathbf{y}) < \mathcal{D}_\infty(\mathbf{x})\} \tag{178}$$

Definition 10.4. The 26-opening map $\Omega_\infty : G_{w,h,d} \to \mathbb{Z}^+$ is the function defined by:

$$\Omega_\infty(\mathbf{x}) := \max\{\mathcal{D}_\infty(\mathbf{y}) \mid \mathbf{y} \in G_{w,h,d} \text{ and } \mathbf{x} \in \mathcal{B}_\infty(\mathbf{y})\} \tag{179}$$

$\Omega_\infty(\mathbf{x})$ is the opening of the voxel $\mathbf{x}$.

For $\mathbf{x} \in E_D$, $\Omega_\infty(\mathbf{x})$ is the radius of the largest maximal ball inscribed in E_D and containing the voxel $\mathbf{x}$. Consequently, it gives us the size of the maximal ball that can be put in empty space while still covering $\mathbf{x}$, an information that describes the local thickness of empty space around $\mathbf{x}$. Looking at variations in the opening map, we can identify narrow regions connecting large ones. Such regions are generally difficult to traverse using local ray sampling and the following sections aim at building an efficient way of doing so.

10.3.2 *The opening labeling*

We first define a simple labeling based on flat regions of the opening map and illustrate its limitations for our problem.

10.3.2.1 *Definitions*

Definition 10.5. The *opening region* $\mathcal{R}(\mathbf{x}) \subset E_D$ of the voxel $\mathbf{x} \in E_D$ is defined as the maximal 6-connected set of voxels containing $\mathbf{x}$ such that $\forall \mathbf{y} \in \mathcal{R}(\mathbf{x})$, $\Omega_\infty(\mathbf{x}) = \Omega_\infty(\mathbf{y})$.

The region $\mathcal{R}(\mathbf{x})$ is a connected subset of E_D of constant opening. We denote by R_Ω the set of all opening regions of E_D.

Definition 10.6. Let $L : R_\Omega \to \{1, ..., |R_\Omega|\}$ be a function mapping each opening region to a label. The *opening labeling* of E_D with respect to L is the map $L_\Omega : E_D \to \{1, ..., |R_\Omega|\}$ defined by:

$$L_\Omega(\mathbf{x}) := L(\mathcal{R}(\mathbf{x})) \tag{180}$$

10.3.2.2 *Limitations*

We aim at tracing rays that travel the scene to reach regions containing a light source or the camera. More specifically, we want to be able to traverse efficiently narrow regions, such that sampled paths do not remain stuck in a region containing no feature. As shown in Figure 81, the labeling L_Ω produces false positive regions for our purpose. Regions like A are a problem because they lead directly on a wall. Ideally we want region A to be part of region B. Our experiments show that all voxels of A are contained in maximal balls of region B. We use this information in section 10.3.3 to define our new labeling.

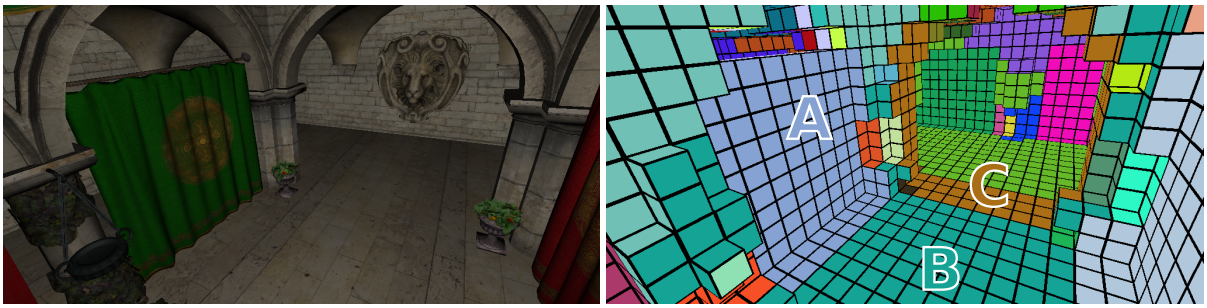


Figure 81 – The opening labeling of a scene. Region A is a false positive because going from B to A does not grant access to other regions of the scene and leads directly on a wall. Region C is interesting because it represents a passage to enter a corridor. It traduces a hole in the 3D scene.

Finding a good merging criterion that meets our expectation would be difficult and would likely depend on parameters that vary from scene to scene. Instead of merging regions, we decided to develop another labeling method based on maximal balls used to obtain the opening map.

10.3.3 *The opening forest labeling*

As mentioned in section 10.3.2, maximal balls give us an information on the similarity between two neighbor regions. When the union of maximal balls of a region A covers entirely a neighbor region B, it means that it is probably easy to access B from A by local sampling of rays.

The opening of a voxel $\mathbf{x}$ is assigned as the radius of a maximal ball $\mathcal{B}_\infty(\mathbf{y})$ containing it. Therefore, the center $\mathbf{y}$ of the ball can be seen as the parent of $\mathbf{x}$ in a forest that can be computed together with the opening map. The roots of such a forest are voxels perfectly centered in the empty space, i.e. voxels $\mathbf{z}$ such that $\Omega_\infty(\mathbf{z}) = \mathcal{D}_\infty(\mathbf{z})$. By descending on a tree of the forest, we are able to detect voxels with slightly lower opening but easily accessible from the region containing the root of the tree.

Figure 82 illustrates a branch of such forest. The red voxel belongs to a false positive regions because its opening is lower than the opening of its parent. The goal of our new labeling procedure is to use the structure the forest, implicitly defined by the computation of the opening map, to extract better regions.

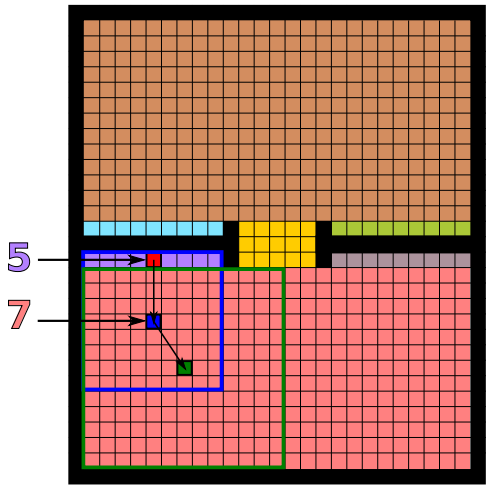


Figure 82 – Illustration of a branch of an opening forest.

Definition 10.7. An *opening forest* of E_D is a map $\Omega_f : E_D \to E_D$ satisfying the following properties:

$$\forall \mathbf{x} \in E_D, \quad \Omega_f(\mathbf{x}) = \mathbf{x} \iff \Omega_\infty(\mathbf{x}) = \mathcal{D}_\infty(\mathbf{x}) \tag{181}$$

$$\forall \mathbf{x} \in E_D, \quad \mathbf{x} \in \mathcal{B}_\infty(\Omega_f(\mathbf{x})) \text{ and } \mathcal{D}_\infty(\Omega_f(\mathbf{x})) = \Omega_\infty(\mathbf{x}) \tag{182}$$

The second properties means that $\Omega_f(\mathbf{x})$ is the center of a maximal ball of radius $\Omega_\infty(\mathbf{x})$ containing $\mathbf{x}$.

There exists several opening forests for a given opening map since a voxel $\mathbf{x}$ can be contained in several maximal balls of radius $\Omega_\infty(\mathbf{x})$. An opening forest can be seen as a structure encoding a possible propagation in E_D to compute the associated opening map. Our goal is to build the regions of our new labeling from the trees of an opening forest. Since we want our regions to be 6-connected, we need to use at least an opening forest such that each tree is 6-connected.

Algorithm 6 computes both the opening map and an opening forest Ω_f that meets this criterion. The 26-distance allows to compute efficiently the opening map by performing six scans over the distance map in each of the six directions (north, south, est, west, top and

bottom) [Vin94]. Each scan performs an independent dilation of each line of the 26-distance map and can be easily implemented in parallel.

Algorithm 6 : Computes the opening map and the opening forest

Data : The distance map $\mathcal{D}_\infty$
Result : The opening map Ω_∞ and the opening forest Ω_f

$\Omega_\infty \leftarrow \mathcal{D}_\infty$
for $(i,j,k) \in [\![1,w]\!] \times [\![1,h]\!] \times [\![1,d]\!]$ **do**
  $\Omega_f(i,j,k) \leftarrow (i,j,k)$
for $(j,k) \in [\![1,h]\!] \times [\![1,d]\!]$ **do**
  dilateLine($\Omega_\infty([1...w],j,k)$, $\Omega_f([1...w],j,k)$)
  dilateLine($\Omega_\infty([w...1],j,k)$, $\Omega_f([w...1],j,k)$)
for $(i,k) \in [\![1,w]\!] \times [\![1,d]\!]$ **do**
  dilateLine($\Omega_\infty(i,[w...h],k)$, $\Omega_f(i,[w...h],k)$)
  dilateLine($\Omega_\infty(i,[h...1],k)$, $\Omega_f(i,[h...1],k)$)
for $(i,j) \in [\![1,w]\!] \times [\![1,h]\!]$ **do**
  dilateLine($\Omega_\infty(i,j,[1...k])$, $\Omega_f(i,j,[1...k])$)
  dilateLine($\Omega_\infty(i,j,[k...1])$, $\Omega_f(i,j,[k...1])$)

Algorithm 7 : dilateLine(R[N], C[N])

Data : R: a line of N radius values, C: a line of N voxel centers
Result : R and C are dilated according to the radius values of R

maxballQueue ← EmptyQueue
for $i \leftarrow 0$ *to* $N-1$ **do**
  currentBall ← { index: i, radius: R[i], center: C[i], end: i + R[i] }
  if *currentBall.radius = o* **then**
    maxballQueue ← EmptyQueue
  else if *maxBallQueue not empty and i = maxballQueue.front().end* **then**
    maxballQueue.pop_front()
  if *maxBallQueue is empty* **then**
    maxballQueue.push_back(currentBall)
  else
    if *currentBall.radius* $\geq$ *maxballQueue.front().radius* **then**
      maxballQueue ← EmptyQueue
      maxballQueue.push_back(currentBall)
    else
      while *currentBall.radius* $\geq$*maxballQueue.back().radius* **do**
        maxballQueue.pop_back()
      if *maxballQueue.back().end* $<$ *currentBall.end* **then**
        maxballQueue.push_back(currentBall)
  $R[i] \leftarrow$ maxballQueue.front().radius
  $C[i] \leftarrow$ maxballQueue.front().center

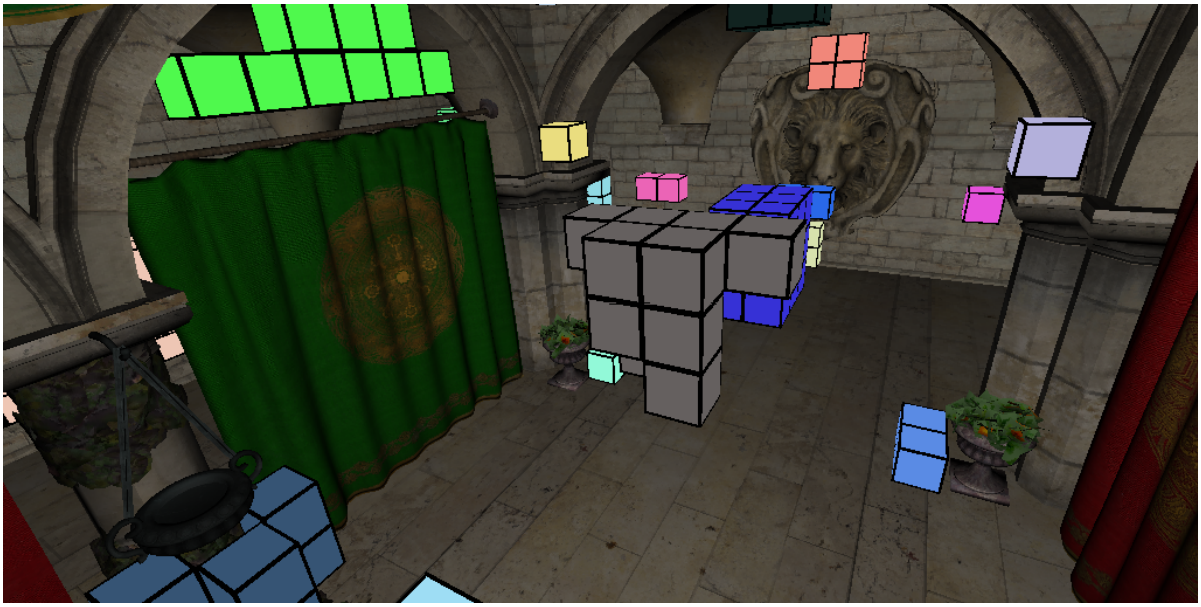


Figure 83 – Some region roots of a scene. Each region is a connected set of centered voxels of the same opening.

We use the opening forest to build a new labeling of the empty space E_{D}. First we define the notion of *region root*.

Definition 10.8. Let $\mathbf{x} \in E_{\mathrm{D}}$ such that $\Omega_f(\mathbf{x}) = \mathbf{x}$. The *region root* $\mathcal{R}_{root}(\mathbf{x}) \subset E_{\mathrm{D}}$ is the maximal 6-connected set of voxels containing $\mathbf{x}$ such that $\forall \mathbf{y} \in \mathcal{R}_{root}(\mathbf{x})$ we have $\Omega_f(\mathbf{y}) = \mathbf{y}$ and $\Omega_\infty(\mathbf{x}) = \Omega_\infty(\mathbf{y})$.

All voxels of a region root have the same opening and are roots of the opening forest. Figure 83 illustrates this concept.

Let R_{root} denotes the set of all region roots of E_{D}. The *opening forest labeling* is built from the opening forest and region roots.

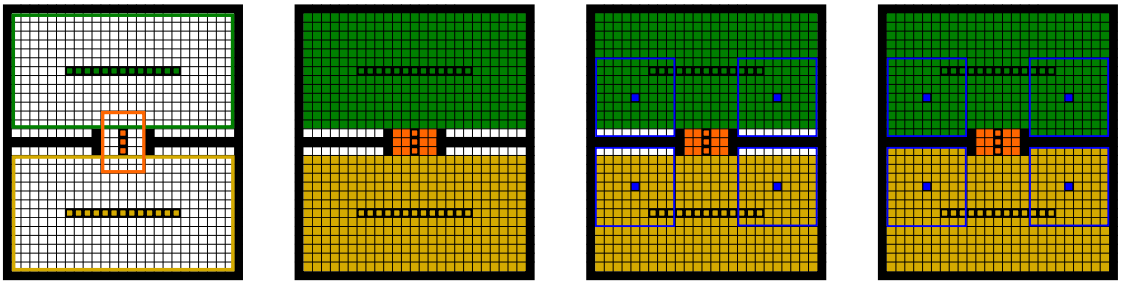
Definition 10.9. Let $L : R_{root} \to \{1, ..., |R_{root}|\}$ be a function mapping each region root to a label. The opening forest label $L_f(\mathbf{x})$ with respect to L of the voxel $\mathbf{x}$ is defined recursively by:

$$\begin{cases} L_f(\mathbf{x}) = L(\mathcal{R}_{root}(\mathbf{x})) \text{ if } \Omega_f(\mathbf{x}) = \mathbf{x} \\ L_f(\mathbf{x}) = L_f(\Omega_f(\mathbf{x})) \text{ otherwise.} \end{cases} \tag{183}$$

The opening forest labeling is a propagation of the label of a region root to the trees rooted in that region (illustrated in Figure 84).

Definition 10.10. The opening forest region $\mathcal{R}_f(\mathbf{x})$ of a voxel $\mathbf{x}$ is the maximal set of voxels such that $\forall \mathbf{y} \in \mathcal{R}_f(\mathbf{x}), L_f(\mathbf{x}) = L_f(\mathbf{y})$. If each tree of the opening forest is 6-connected, then this set is 6-connected.

We denote by R_f the set of all opening forest regions of E_{D}.



(a) Regions roots and union of their maximal balls (b) Propagation (c) Maxballs spanning unlabeled voxels (d) Final labeling

Figure 84 – Computation of the opening forest labeling.

10.3.4 *Opening portals*

We now introduce *opening portals*, defined as 2D-surfaces separating opening forest regions. Since opening portals are not composed of voxels, we must define them as sets of 2-faces, already introduced in Section 5.2.1 covering the cubical complex framework.

Let $X, Y \in R_f$ be two opening forest regions of E_{D}. We say that X and Y are neighbor regions if $\exists \mathbf{x} \in X, \exists \mathbf{y} \in Y$ such that $\mathbf{x}$ and $\mathbf{y}$ are 6-neighbors.

Recall that the 3-face associated to a voxel $\mathbf{x} \in \mathbb{Z}^3$ is $\Psi(\mathbf{x}) := \{x_1, x_1+1\} \times \{x_2, x_2+1\} \times \{x_3, x_3+1\}$ and corresponds to its eight corners. This notion allows to define the two dimensional face separating two neighbor voxels.

Definition 10.11. Let $\mathbf{x}$, $\mathbf{y}$ be two 6-adjacent voxels. The set $F_s(\mathbf{x}, \mathbf{y}) = \Psi(\mathbf{x}) \cap \Psi(\mathbf{y})$ is the 2-face that separates $\mathbf{x}$ and $\mathbf{y}$. It is composed of four points.

Definition 10.12. Let $X, Y \in R_f$ be two neighbor opening forest regions of E_{D}. The opening portal $P_{X,Y}$ separating X and Y is the set defined by:

$$f \in P_{X,Y} \Leftrightarrow \exists \mathbf{x} \in X, \exists \mathbf{y} \in Y \text{ such that } f = F_s(\mathbf{x}, \mathbf{y}) \tag{184}$$

We can build rays going from X to Y by sampling points on the 2-faces of $P_{X,Y}$.

10.3.5 *Results*

We present the result of our *opening forest labeling* on different scenes and for different resolutions of the voxelization. To illustrate regions, we display the voxelization of the scene such that the color of a face f of each voxel $\mathbf{x} \in S$ identifies the label of the empty-space voxel $\mathbf{y} \in E$ which is adjacent to $\mathbf{x}$ for the face f ($f = F_s(\mathbf{x}, \mathbf{y})$). We also show opening portals separating different regions and we compare a local ray sampling strategy (uniform sampling of directions on the hemisphere of the origin point) to the a strategy that samples rays passing through our opening portals.

10.3.5.1 *Ajar door scene*

Figure 85 illustrates the opening forest labeling of the Ajar door scene presented in Section 10.1. Figure 86 shows opening portals and demonstrate that our sampling strategy efficiently samples rays traversing the door's aperture or leaving the region located under the table.

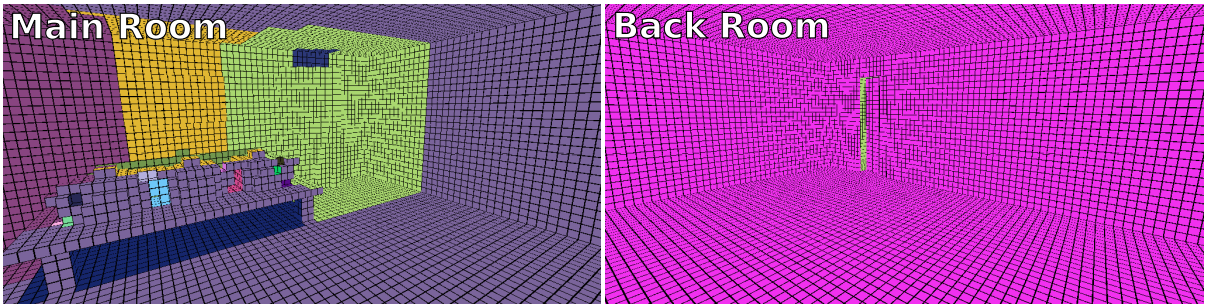


Figure 85 – Our opening forest labeling for the Ajar Door scene and a grid resolution of (72, 118, 28). We observe that the back room is composed of one large region (right) separated from the main room by the narrow door's aperture, as expected. The main room (left) is split in several regions allowing to travel the scene efficiently by sampling portals (see Figure 86).

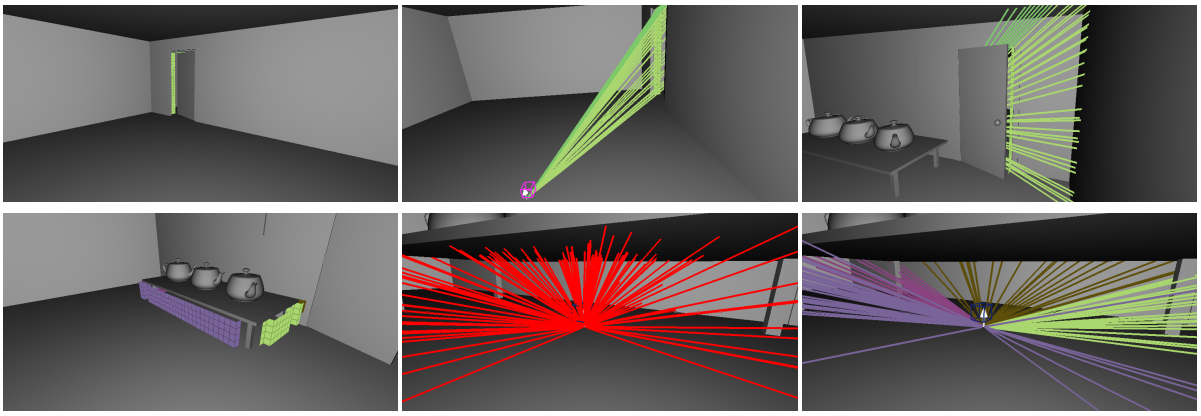


Figure 86 – *Top-left*: the green portal separates the back room from the main room. *Top-center*: 128 rays are sampled through the portal, starting at a point of the back room. *Top-right*: we observe that all rays reach the main room. *Bottom-left*: portals separating the region under the table from neighbor regions. *Bottom-center*: rays sampled with the local strategy. Many of them hit the back of the table. *Bottom-right*: with our strategy, all sampled rays leave the region.

10.3.5.2 *Sibenik scene*

Figure 87 and 88 illustrate our method on the Sibenik scene, which represents a church. This scene features less occlusions than the two others. Nevertheless, we demonstrate that we can use our sampling strategy to pass through a specific portal in order to reach a particular area of the scene. Being able to do this is useful to reach a specific light source after few reflections.

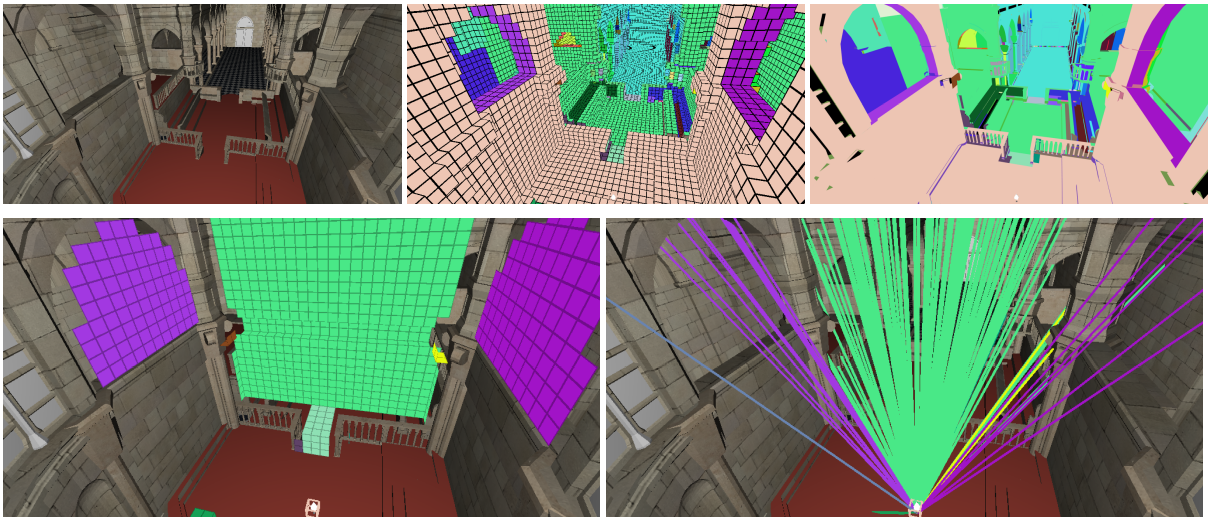


Figure 87 – Top row shows the labeling of the Sibenik scene for a grid resolution of (138, 104, 58). The top-right image is the labeling shown directly on the surfaces of the scene. The bottom-left image demonstrates three major portals of a region. The bottom-right image illustrates the sampling of rays leaving that region by selecting portals based on their area (more rays are sampled on large portals).

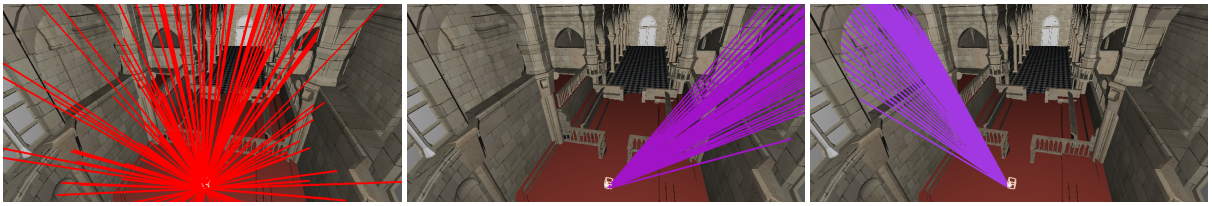


Figure 88 – Left image shows the local sampling of rays which is not efficient to pass through the left and right portal. Using our method, it is straightforward to sample rays through a specific portal as pointed out in the middle and right image.

10.3.5.3 *Sponza scene*

Figure 89 and 90 show results of our method on the Sponza scene. This scene is composed of several corridors occluded from the main part of the scene by drapes. Starting from a corridor, reaching the main part is hard due to narrow exits. Our sampling strategy enables to do it efficiently. Figure 90 demonstrates the robustness of our method regarding the resolution of the voxel grid. Opening portals remain stable and fit better to the scene as we increase the resolution.

10.3.5.4 *Limitations*

Our new labeling still produces neighbor regions with close opening and highly connected by their maximal balls. Such regions are separated because their region roots are not connected. However, these regions exposes better coherency with their opening: a region having a region root with high opening has a high volume and allows to access all of its narrow neighbors using small portals.

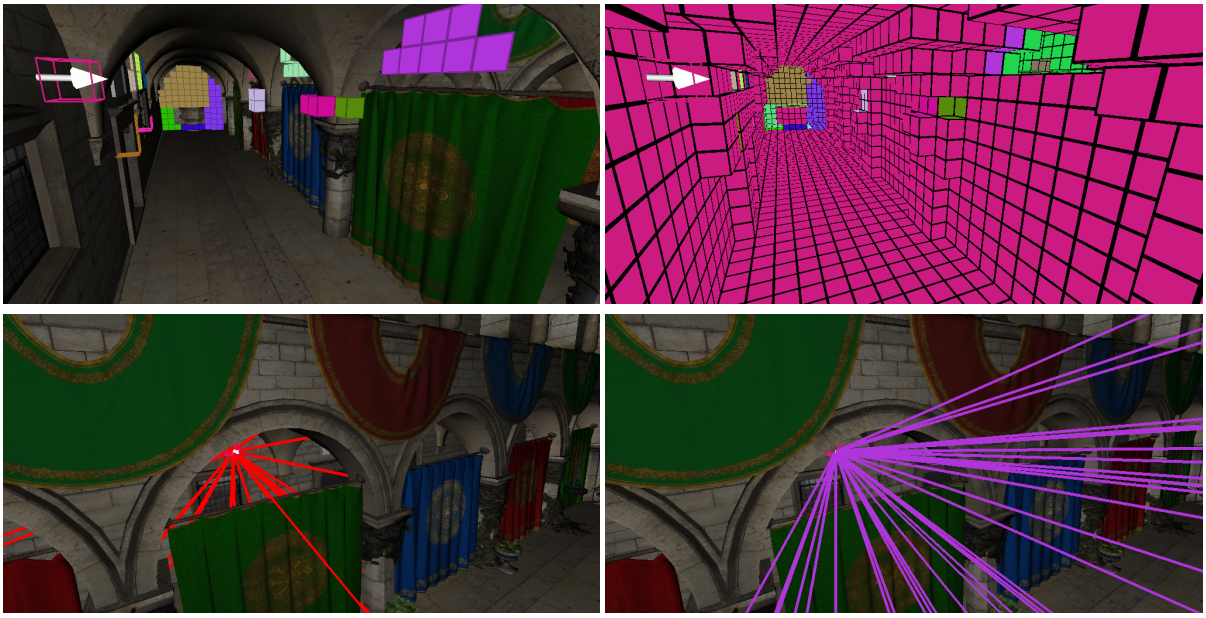


Figure 89 – Top row: portals and labeling of a corridor of the Sponza scene for a resolution of (118, 50, 72). Bottom row: comparison of local sampling (left) and sampling through a chosen portal (right), allowing to leave efficiently the corridor.

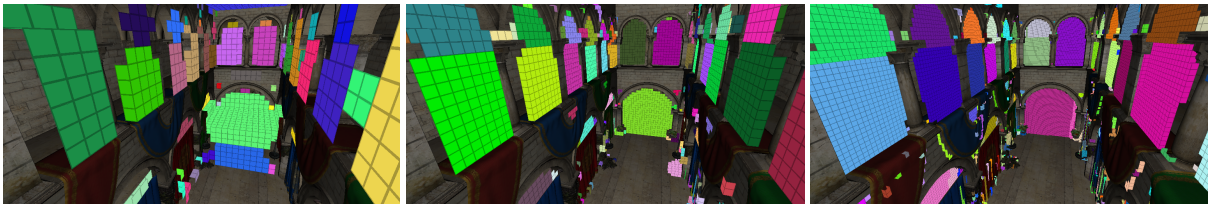


Figure 90 – Illustration of portals as the resolution of the voxel grid increases. We observe that the separation between regions becomes more precise as we increase the resolution of the voxel grid.

A more important issue regarding the sampling of rays is the presence of some concave regions. In such a region and without more information, we could sample a portal which is not visible from the origin of the ray. To apply our method to light transport algorithms, it might be unavoidable to use a convex decomposition algorithm or to improve our algorithm to guaranty convexity of regions.

10.4 FIRST EXPERIMENTS

In this section, we present and discuss our first experiments regarding the usage of our opening portals for path tracing. These experiments are, for now, limited to the Door scene, since it presents the main kind of visibility feature that our strategy targets: a narrow opening that links two large rooms. The illumination setting is the one shown in Figure 78.

We implemented two sampling strategies, and both produce discontinuities in rendered images, due to a fast change between sampling pdfs at neighbor surface points. We discuss the benefits and problems of each strategy, as well as future works to avoid discontinuities.

Since our implementations are not optimized and quite expensive in their current state, we compare them against standard path tracing for the same number of rendering iterations instead of the same rendering time.

10.4.1 *Shortest paths strategy*

Our first strategy is inspired by our ray sampling strategy based on the skeleton, introduced in Chapter 9. Before rendering, we build a shortest path tree on the graph of opening forest regions. Each shortest path lead to the region containing the light source.

Let $\mathbf{x} \in \mathcal{M}$ be a surface point and $\mathcal{R}_f$ its opening forest region. We want to sample a ray going toward bright parts of the scene. For that, we sample a point $\mathbf{y}$ uniformly on the portal separating the region $\mathcal{R}_f$ from the next region in its shortest path. The sampled direction at $\mathbf{x}$ is then set to $\omega_{\mathbf{x},\mathbf{y}}$.

Figure 91 shows rendering results for different maximal path lengths and 4096 rendering iterations. On each column, the first image is a rendering with standard path tracing, using BSDF sampling. The second image (SIS1) shows a biased rendering, using only our strategy. The third image (SIS2) shows an unbiased rendering, where we combine our portal sampling with BSDF sampling using MIS, with the balance heuristic. For the last two images (SIS3, SIS4), we force the use of the portal located at the door opening for all surfaces points, effectively ignoring the opening forest labeling and considering the most interesting portal of the scene only (illustrated in Figure 93a).

For images using all portals (SIS1, SIS2), we clearly see the discontinuity produced by the fast change between sampling strategies at the portal perpendicular to the wall. This portal is illustrated by Figure 92, together with the opening forest labeling. Using MIS is not sufficient to attenuate the discontinuity since it appears at surface points for which neither the portal sampling strategy, nor the BSDF sampling strategy are good to importance sample incident illumination. Nevertheless, we also observe a good reduction of noise at parts of the image where the sampling strategy remains coherent between surface points.

Forcing the use of the portal at the door's opening (illustrated in Figure 93a) allows to keep the same sampling strategy for all surface points. While the discontinuity still appears when not using MIS (SIS3), because many points are not able to see the portal, the use of MIS allows to make it disappear.

The benefits of our sampling strategy are its simplicity and efficiency. Sampling an outgoing ray is fast since it only require uniform sampling of a portal surface, represented by a mesh.

However, this strategy cannot handle illumination details since a portal can be quite large and all points on its surface are treated equally. Only using portals from the opening forest region containing the surface point produces discontinuity, especially at the boundaries of

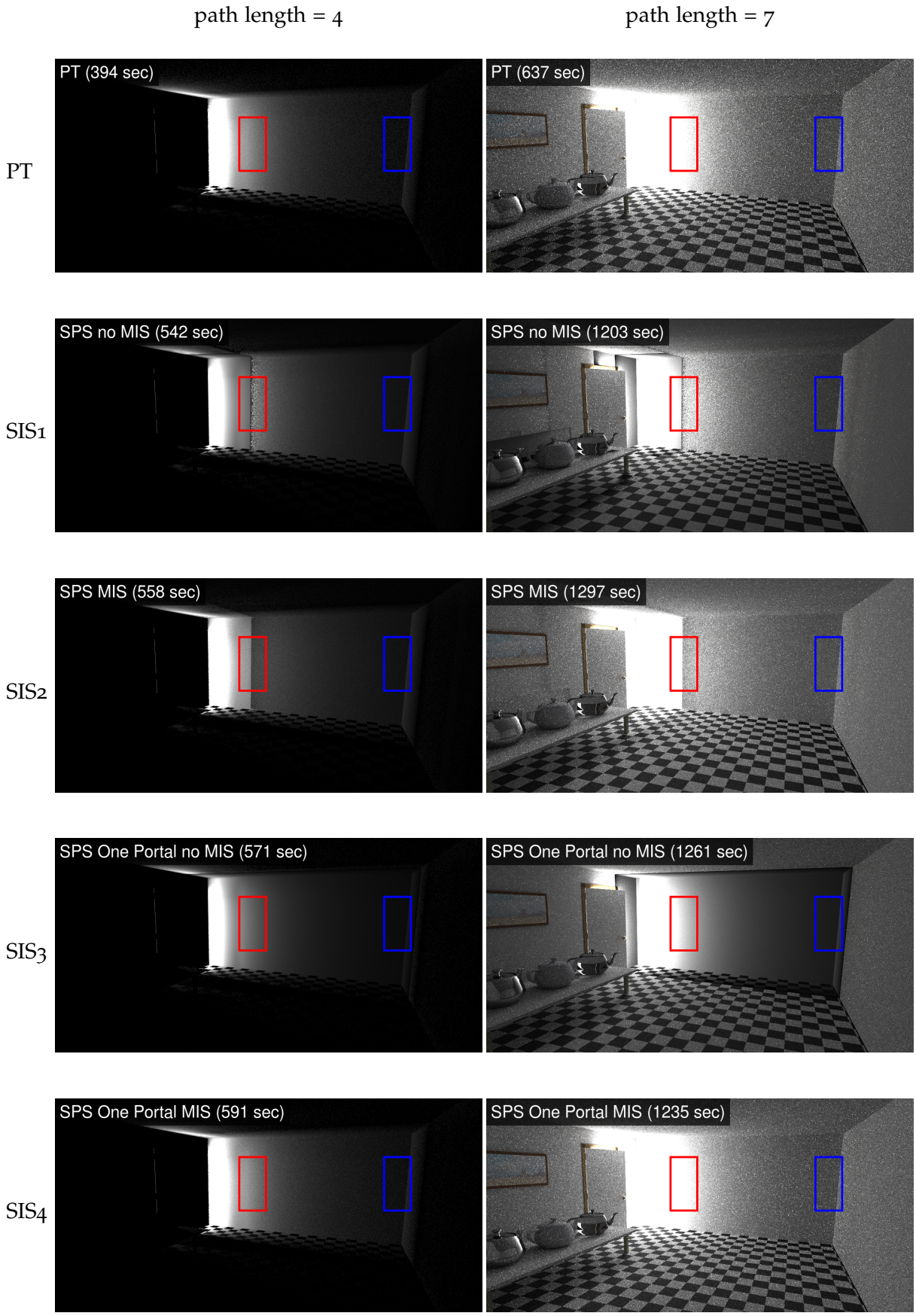


Figure 91 – Shortest paths strategy for 4096 rendering iterations.

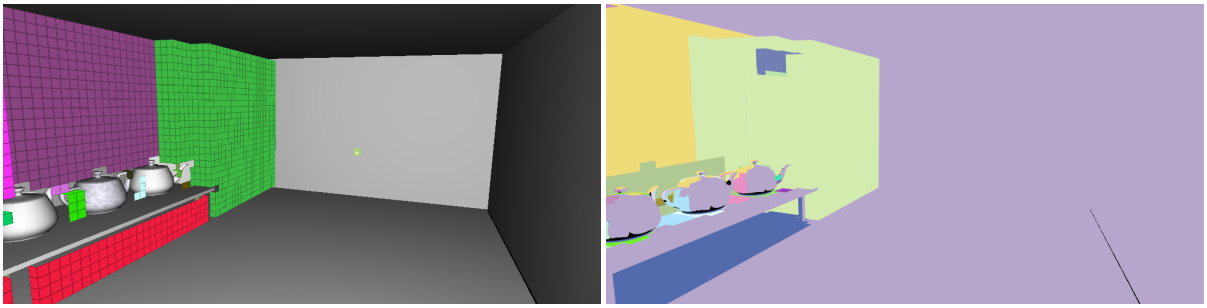
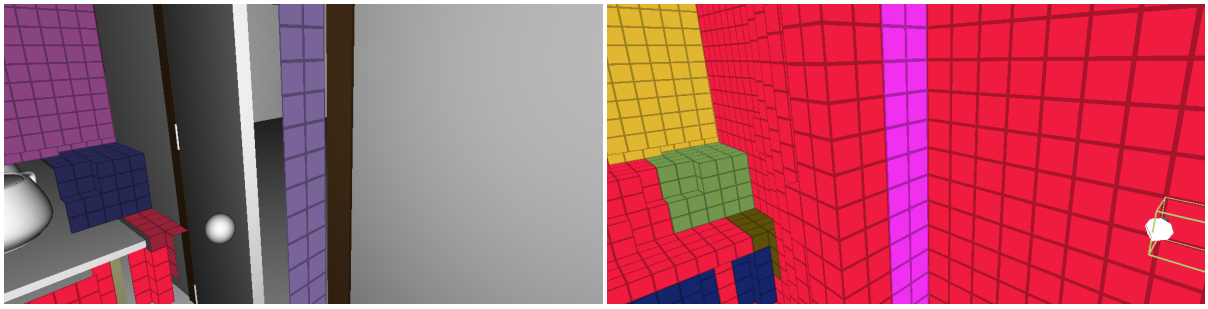


Figure 92 – Opening portals and opening forest labeling. The portals perpendicular to walls are susceptible to produce discontinuities in the sampling of outgoing rays.



(a) Door's opening portal. (b) Door's adjacent opening border

Figure 93 – Comparison between opening portals and opening border. The opening border contains all the portals leaving a region, but also faces connected to the scene voxelization. Consequently, it is more robust to sample outgoing directions, without missing those that do not intersect a portal.

the region since portals tend to be perpendicular to walls. A possible solution to this problem would be to also use portals from neighbor regions, based on a similarity criterion. Finally, a portal does not always fit the complete interface where rays can pass, as illustrated by Figure 93a. This is actually a huge problem regarding the use of MIS to combine our strategy with BSDF sampling, since such rays, missing the portal, cannot be generated by our method and do not benefit from MIS weighting. The contributions of paths containing these kind of rays is estimated only with the BSDF sampling strategy and appear as fireflies in images rendered with few iterations.

10.4.2 *Face irradiance strategy*

Our second sampling strategy aims at solving two problems of the previous one: handling illumination details and avoiding missing some high contribution rays. For that, we do not treat portals, but individual 2D faces separating an opening forest region from its neighbors. We also consider faces separating a region from the scene voxelization, in order to give a chance of being sampled to each direction leaving a region. The set of all faces separating a region from neighbors is referred as the *opening border* of that region (illustrated in Figure 93b).

Before each rendering iteration, we estimate the irradiance at each face with one path, obtained with standard BSDF sampling. From these estimations, a discrete probability distribution is built on all faces of the opening border of each region (each region has its own distribution).

Let $\mathbf{x} \in \mathcal{M}$ be a surface point and $\mathcal{R}_f$ its opening forest region. To sample an outgoing ray at $\mathbf{x}$, we randomly chose a face on the opening border of $\mathcal{R}_f$, according to the precomputed discrete probability distribution, and we sample uniformly a point $\mathbf{y}$ on that face. The sampled direction at $\mathbf{x}$ is then $\omega_{\mathbf{x},\mathbf{y}}$.

In order to attempt the removal of discontinuities, we also introduce two parameters: a distance threshold d_t and an orientation threshold o_t. If the distance between $\mathbf{x}$ and the center of the chosen face is larger that d_t, we cancel the selection and randomly chose a face on the opening border of the neighbor opening forest region that is connected to the originally selected face. We do the same if the cosine between the normal of the face and the direction between the center of the face and $\mathbf{x}$ is greater than o_t. This strategy gives an opportunity of using a better opening border if the one of the region of the point is not well adapted (the point is too close, or located at a grazing angle).

Figure 94 shows rendering results for different maximal path lengths and 4096 rendering iterations. While this sampling strategy still produces discontinuities (FIS1, FIS2), they are better smoothed than the shortest paths strategy because we allow using faces from neighbor regions. For these results, we use the values $d_t = 0.01 \times d$, where d is the diagonal of the bounding box of the scene, and $o_t = 0.5$. The last two results (FIS3, FIS4) are limited to sampling faces from the opening border of the region adjacent to the door (illustrated in Figure 93b).

This second strategy has several advantages over our previous one:

— It is general and does not depends on the type or number of light sources.
— It handles details in spatial illumination (but depends on the voxelization resolution).
— Any direction leaving a region can be sampled, assuming the irradiance estimation give it a contribution at some point.
— The irradiance estimation of each face can be done while rendering the image, by accumulating contributions of paths intersecting a face.

Our current implementation of this strategy is not optimized and too much expensive, which may offset its advantages. We also suspect a bug in our implementation because our results are slightly brighter than a reference image (it might be due to the use of unormalized pdfs). The sampling strategy is also unable to capture details in the directional distribution of illumination, since it is based on irradiance at each face. Finally, discontinuities are still too important to be ignored, and further work is required to make the sampling strategy reliable.

10.5 CONCLUSION AND PERSPECTIVES

We presented a new labeling method to partition the empty space of a 3D scene according to opening and maximal balls. This labeling is used to extract portals between 3D regions

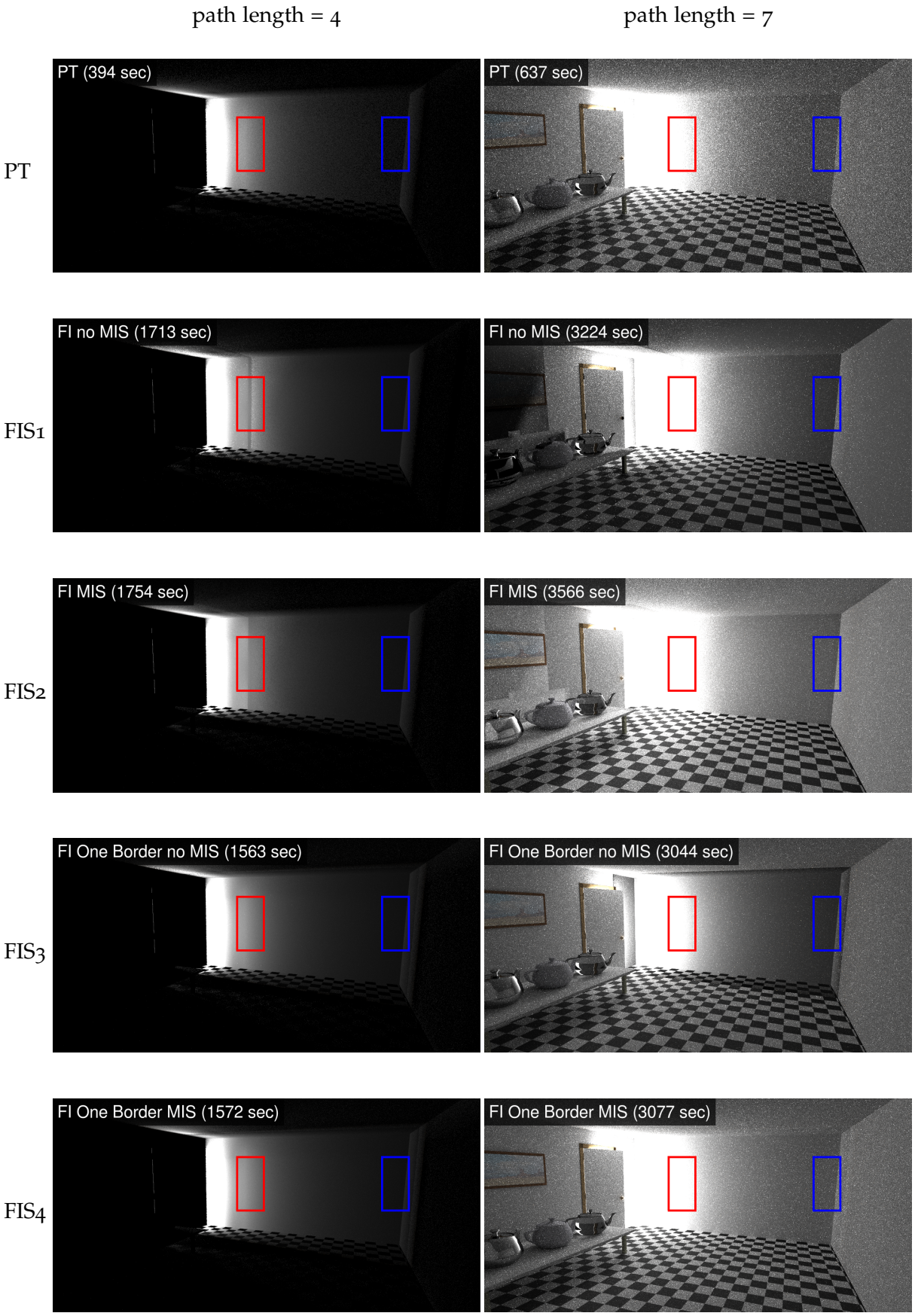


Figure 94 – Face irradiance strategy for 4096 rendering iterations.

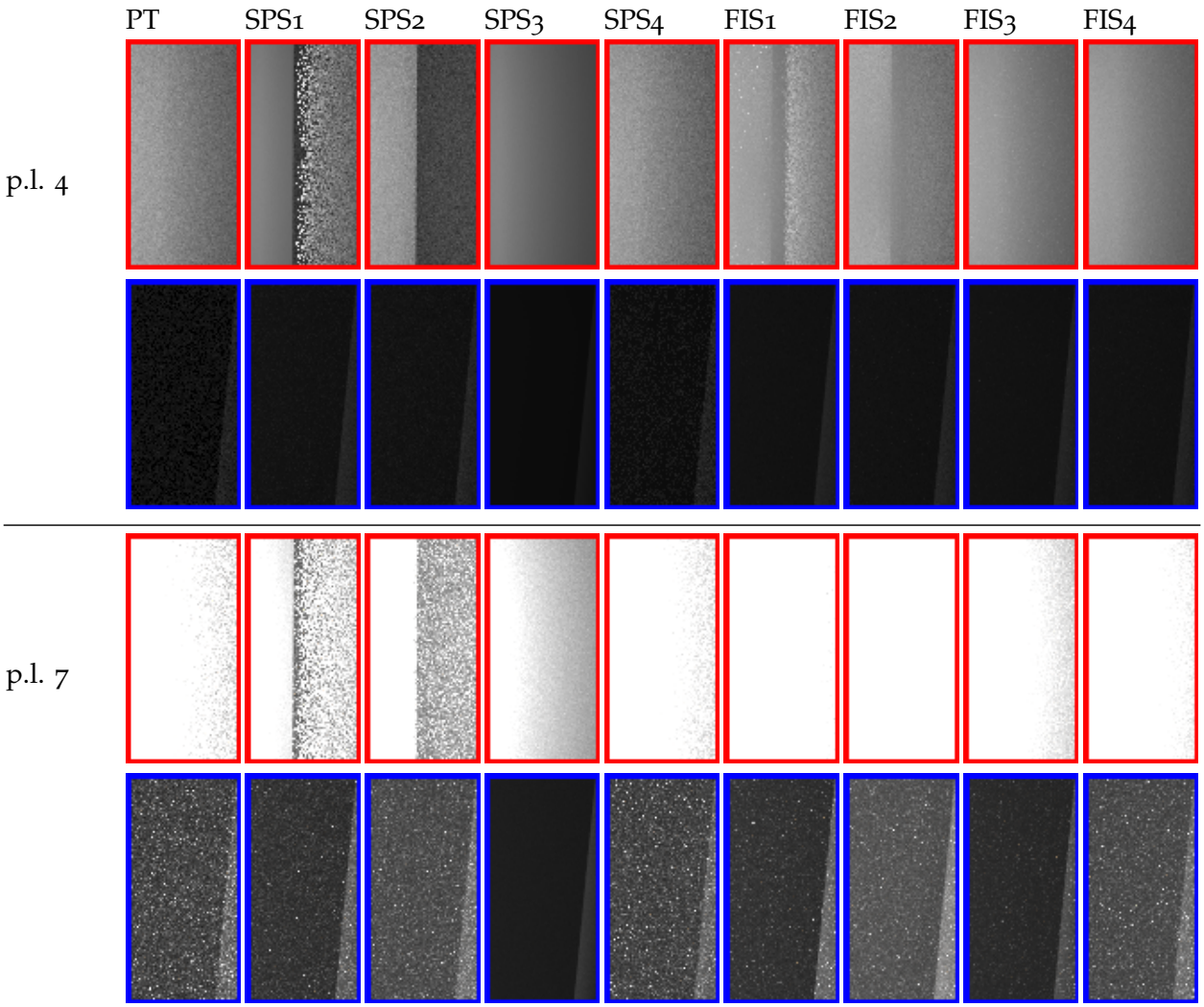


Figure 95 – Zoom on the presented results, presenting a discontinuity (red frame) and noise reduction (blue frame). While the discontinuity is too important to be ignored, our strategies are able to reduce noise at parts of the image where the sampling is smooth.

that can be efficiently sampled to go from one region to another when tracing rays in the 3D scene.

LABELING On future works, we would want to explore other labeling methods in order to compare them with our on the application to ray sampling. For example, our region roots can be used as seeds for a watershed propagation on the opening map instead of using the opening forest to define the labeling. Comparing the resulting labeling to our opening forest labeling could be interesting to take advantage of the strengths of each one. More importantly, we aim at enhancing our method to obtain approximately convex regions. As mentioned in section 10.3.5, not dealing with convex regions can be a problem because sampling points on a portal does not guaranty their visibility from inside the region.

We also plan a theoretical analysis of the properties of our labeling and how they guaranty that portals are well placed regarding the partitioning of the scene in narrow versus large regions. Some portals generated by our method remain useless for ray sampling since the local sampling strategy based on the BSDF is already able to traverse them efficiently. Studying more deeply mathematical properties of the labeling could exhibit a way of getting rid of these portals automatically, by changing the opening forest labeling definition.

RENDERING The two presented sampling strategies produce discontinuities in rendered images, which have to be resolved in order to develop a robust rendering algorithm. These discontinuities are mainly generated because of portals that separate opening forest regions of similar thickness. Consequently, it would be worth working on the labeling method in order to avoid the generation of these portals. Another possibility is to address the MIS weighting strategy, at rendering time, by punishing with a low weight directions that are sampled but not well adapted to the illumination. We observed that discontinuities occur at the separation between two regions. By pre-computing information on the proximity of voxels to the border of the region, it would be possible to combine the ray sampling strategy of the current region with neighbor's ones. We tried that with our thresholding parameters, but a solution based on fixed parameters is clearly unreliable for general scene. Overall, our opening based path tracing offers good noise reduction on the presented configuration but requires further improvement to provide robust and efficient rendering on any scene

The method presented in this chapter shares many similarities with our real-time rendering algorithm detailed in Chapter 7. Both methods aims at defining a partitioning of the scene and portals separating regions in order to improve rendering algorithms. In this chapter, we used the opening map instead of the skeleton to achieve such goal, giving more robust portals and regions. One future work is to adapt the method of Chapter 7 to use the structures defined in this chapter and to evaluate how much they can improve real-time rendering with many-lights.

11

SKELETON BASED VERTEX CONNECTION RESAMPLING FOR BIDIRECTIONAL PATH TRACING

Figure 96 – A comparison between standard BPT and BPT improved with our connection resampling strategy (after 60 seconds of rendering). In this setting, the draperies occlude most of the light vertices from the eye vertices, resulting in many null contributions. Our resampling favors connections that are more likely to produce a high contribution.

As mentioned in Section 4.3.2.2, the most expensive operation performed by BPT is the connection between a large number of pairs of vertices, requiring many visibility tests. In highly occluded scenes, most tests fail and present no contribution for the visibility tests cost. For such configurations, a caching and resampling scheme similar to the one proposed by Georgiev et al. [GKPS12] can significantly improve the performance of BPT by favoring connections that are likely to produce a high contribution to the final image. These kinds of strategies generally use surface points as importance caches. For robust resampling, a high cache density is required, inducing a high overhead, both in terms of computation and memory.

In this chapter, we propose a new solution for resampling connections, based on the filtered skeleton of the empty space of the scene introduced in Section 6.3. We use this skeleton to quickly pre-compute discrete probability distributions of light subpaths based on their contribution around the different parts of the skeleton. These distributions are then used while tracing eye subpaths to resample efficiently the connections with light subpaths. Our method can be implemented on top of any bidirectional path tracer, potentially improving any other method based on BPT for sampling paths [GKDS12, VG97, VKŠ+14]. This work has been presented at the PG 2015 conference [NB15] and the source code is freely available on Github [Noë15]. The main contributions presented in this chapter are:

— A new unbiased strategy for vertex connection resampling, based on a pre-computed skeleton of the empty space of the scene.
— A comparison between our method, standard BPT and an adaptation of the method from Georgiev et al. [GKPS12] to BPT. While the latter algorithm performs better for

the same number of rendering iterations, our method gives better results for the same rendering time thanks to a smaller overhead.

Contents

11.1 MOTIVATION AND OVERVIEW

Connecting two vertices is an expensive operation because it involves a visibility test. For scenes with complex visibility, it can be very inefficient since many such tests result in no contribution. Indeed, blindly connecting random vertices does not importance sample the visibility function and can introduce high variance in the estimation. Resampling is a general method that applies Monte Carlo estimation to the sum of contributions already sampled for Monte Carlo integration (see Section 3.3.4). It can be used to reduce the number of connections that must be evaluated and to increase the probability of connecting vertices that are likely to produce a high contribution.

Talbot et al. [TCE05] investigate the use of resampling for Monte Carlo rendering. Their method can be applied to the connection problem in BPT and many-light methods by building a discrete probability distribution over the light vertices for each eye vertex. This distribution is built using the contribution of each light vertex without the visibility factor. This operation is expensive when the number of light vertices is high. Moreover, ignoring visibility reduces the robustness of the method in highly occluded scenes since arbitrary high contributions can be reduced to zero when multiplied by the visibility factor.

Georgiev et al. [GKPS12] introduce the importance caching (IC) method and apply the same idea to many-light rendering. However, they build discrete resampling distributions that include the visibility factor on a sparse set of surface points called importance records. For each point to illuminate, they gather nearest importance records using a Kd-tree and resample VPLs according to the pre-computed distributions for these records. By combining these distributions with more conservative ones, they propose a robust estimator that is able to deal with complex scenes. However, gathering nearest importance records is expensive and increases the rendering time dramatically when applied to BPT. Indeed, the number of eye vertices is higher for BPT due to multiple reflections along eye subpaths, and thus increases the number of nearest neighbor queries. Moreover, more resampling distributions have to be pre-computed before each rendering iteration since importance records must be spread on more surfaces than only those visible from the camera. Finally, BPT already importance samples efficiently the directional component of the BSDF through local eye subpath sampling, which reduces the interest of introducing it in resampling distributions.

As opposed to established methods, our algorithm caches resampling distributions on a sparse set of points centered in the empty space of the scene, obtained from a curvilinear skeleton. This approach significantly reduces the number of distributions that must be pre-computed for each rendering iteration, while still exploiting visibility and geometric information for efficient resampling.

In its standard shape, BPT samples an eye sub-path and a light sub-path for each pixel and compute all possible connections between them to obtain its primary estimate. To apply resampling on BPT, many light sub-paths have to be computed before resampling few of them. For efficiency, all these sub-paths are shared between pixels, a strategy similar to many-light rendering that has been used by Pajot et al. [PBPP11] for efficient implementation

of BPT on the GPU. Given a sequence of N light subpaths $(\bar{\mathbf{y}}^j_S = \mathbf{y}^j_1 ... \mathbf{y}^j_S)_{j=1,...,N}$ and a single eye subpath $\bar{\mathbf{z}}_T$, the many-path bidirectional estimator can be written as:

$$\begin{aligned} \hat{I} &:= \sum_{t=0}^{T} \sum_{\substack{s=0 \\ s+t>1}}^{S} \frac{1}{N} \sum_{j=1}^{N} w_{s,t}(\bar{\mathbf{x}}^j_{s,t}) \frac{f^{(i)}(\bar{\mathbf{x}}^j_{s,t})}{p_{s,t}(\bar{\mathbf{x}}^j_{s,t})} \\ &:= \sum_{t=0}^{T} \sum_{\substack{s=0 \\ s+t>1}}^{S} \frac{1}{N} \sum_{j=1}^{N} C(\bar{\mathbf{x}}^j_{s,t}) \end{aligned} \tag{185}$$

where $\bar{\mathbf{x}}^j_{s,t} := \mathbf{y}^j_1 ... \mathbf{y}^j_s \mathbf{z}_t ... \mathbf{z}_1$, $w_{s,t}$ is the MIS weighting function, $f^{(i)}$ is the contribution function and $p_{s,t}$ the bidirectional path sampling strategies (all these quantities are defined in Section 4.1.3).

Connections are performed for $s, t \neq 0$, thus the inner sum can be resampled to give the new estimator:

$$\hat{I}_R := \sum_{t=1}^{T} \sum_{s=1}^{S} \frac{1}{N} \frac{C(\bar{\mathbf{X}}_{s,t})}{p^R_{s,t}(\bar{\mathbf{Y}}_{s,t})} \tag{186}$$

Where $\bar{\mathbf{X}}_{s,t}$ is formed by connecting a resampled light subpath $\bar{\mathbf{Y}}_{s,t}$ with the eye subpath $\bar{\mathbf{z}}_t$. The light subpath $\bar{\mathbf{Y}}_{s,t}$ is obtained according to a discrete probability distribution $p^R_{s,t}$ associated to the eye vertex $\mathbf{z}_t$.

Finding good resampling distributions is challenging because a tradeoff must be made between resampling quality and computation speed. Contrary to previous resampling approaches, our distributions are not cached at surface points but at skeleton nodes, which are points centered in the empty space of the scene. We build our distributions from visibility between nodes and light vertices, but also distance and partial contribution associated to each light vertex. Caching this kind of information at points of empty space instead of surface points is motivated by several arguments:

— Our skeleton sparsely covers the entire scene, therefore few resampling distributions must be pre-computed compared to the number of vertices involved in the estimation.
— The similarity between the visibility function of a node and the visibility function of eye vertices mapped to the node is enough for coarse but cheap resampling.
— Each resampling distribution is shared by a high number of eye vertices, inducing more coherent resampling between these vertices.
— Thanks to our node mapping grid, we obtain in constant time the distributions stored at the nearest node of each eye vertex.

As shown in Section 11.5, our method improves the convergence speed of bidirectional path tracing at a negligible cost. Moreover, the algorithm is simple to implement on top of any BPT implementation.

Our algorithm progressively renders the image by performing the following steps at each iteration:

1. Sample N light subpaths, resulting in $N \times S$ light vertices where S is the maximal length of a light subpath.

2. Build the resampling distributions of light vertices for each node of the skeleton.
3. Sample an eye subpath for each pixel and estimate the measurement integral by resampling light vertices according to the resampling distributions.

The first step is identical to light subpath tracing performed for bidirectional path tracing, but we store light vertices to resample them later. We detail the second and third step in Section 11.2 and Section 11.3 respectively.

11.2 SKELETON NODE RESAMPLING DISTRIBUTIONS

Each node stores $3 \times S$ discrete cumulative distribution functions (CDFs) of size N to perform robust and unbiased resampling. Each CDF is restricted to light subpaths having the same length, in order to connect each eye vertex to one light subpath of each length. The three distributions for a given length are combined using multiple importance sampling at the next step. This approach is similar to importance caching [GKPS12] in which they use four distributions at each importance record.

Our first and second distributions both depend on a common weighting function. Let $\mathbf{n}$ be a node of the skeleton, with position $P_{\mathbf{n}}$. For each light vertex $\mathbf{y}_s^i$, we define the following weight:

$$w_{\mathbf{n}}(\mathbf{y}_s^i) := \frac{L(\mathbf{y}_s^i)}{||P_{\mathbf{n}} - \mathbf{y}_s^i||^2} \tag{187}$$

where $L(\mathbf{y}_s^i)$ is the partial contribution of the subpath $\bar{\mathbf{y}}_s^i$, that is, the product of emitted radiance, geometric factors and scattering factors divided by the probability density of sampling the subpath.

Our weights are proportional to the inverse squared distance between the node and light vertices to favor those located close to the node. This choice is driven by the fact that such light vertices are also located close to eye vertices mapped to the node. Since the geometric factor between vertices also depends on the inverse squared distance, our weights tend to favor light vertices that give a high geometric factor when connected with an eye vertex.

Our first distribution combines the weighting function (187) with the visibility function of the node:

$$p_{V,\mathbf{n}}(\mathbf{y}_s^i) = \frac{V(\mathbf{y}_s^i, P_{\mathbf{n}}).w_{\mathbf{n}}(\mathbf{y}_s^i)}{\sum_{j=1}^{N} V(\mathbf{y}_s^j, P_{\mathbf{n}}).w_{\mathbf{n}}(\mathbf{y}_s^j)} \tag{188}$$

By favoring light vertices visible from the node, we ensure a good resampling for all eye vertices that share many visible points with their associated node, which is the case for most of them. However, this distribution is biased since it gives zero probability to some light vertices that actually contribute to eye vertices mapped to the node.

Our second distribution is similar to the first but does not take into account visibility:

$$p_{U,\mathbf{n}}(\mathbf{y}_s^i) = \frac{w_{\mathbf{n}}(\mathbf{y}_s^i)}{\sum_{j=1}^{N} w_{\mathbf{n}}(\mathbf{y}_s^j)} \tag{189}$$

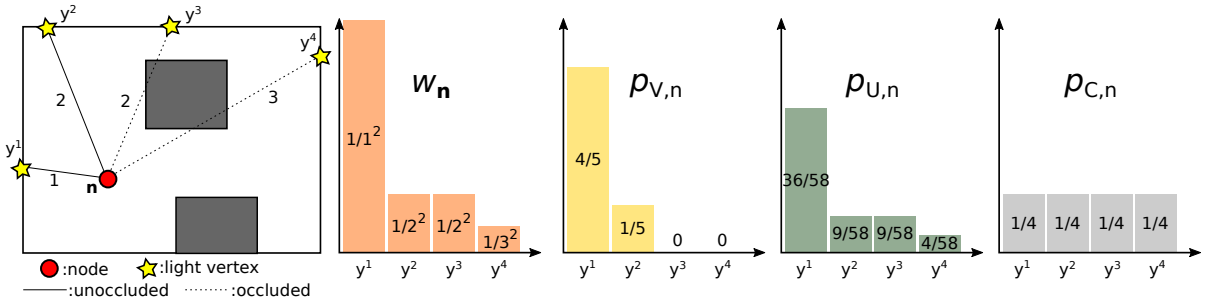


Figure 97 – An illustration of the node weighting function $w_{\mathbf{n}}$ and the three resampling probability distributions $p_{X,\mathbf{n}}$, $X \in \{V, U, C\}$ for a node $\mathbf{n}$ and four light vertices $\mathbf{y}^i$. For simplicity, we suppose here that $L(\mathbf{y}^i) = 1, i = 1...4$, thus we have $w_{\mathbf{n}}(\mathbf{y}^i) = \frac{1}{||P_{\mathbf{n}} - \mathbf{y}^i||^2}$. Distances from the node to light vertices are shown on the scene illustration.

This one is important for eye vertices mapped to a node that does not approximate well their visibility function. It also ensures that our estimator is unbiased.

Sampling a distant light vertex with distribution $p_{U,\mathbf{n}}$ would give it low probability and possibly introduce additional variance if the light vertex contributes to the illumination of some eye vertices mapped to the node. To avoid this, we use a third resampling distribution, uniform among light subpaths of the same length:

$$p_{C,\mathbf{n}}(\mathbf{y}_s^i) = \frac{1}{N} \tag{190}$$

Figure 97 illustrates both the weighting function and the three resampling probability distributions on a simple example scene.

11.3 EYE SUBPATH SAMPLING

For this last step we sample an eye subpath $\bar{\mathbf{z}}_T = \mathbf{z}_1...\mathbf{z}_T$ through each pixel to estimate the measurement integral using our resampling distributions. We note $\mathbf{n}_t$ the node associated to the eye vertex $\mathbf{z}_t$. It is obtained by looking at the voxel containing $\mathbf{z}_t$ in the 3D-grid that map each voxel to a skeleton node.

The estimator is then:

$$\hat{I}_{Skel} := \sum_{t=1}^{T} \sum_{s=1}^{S} \frac{1}{N} w_{X,\mathbf{n}_t}(\bar{\mathbf{Y}}_{X,s,t}) \frac{C(\bar{\mathbf{X}}_{X,s,t})}{p_X \times p_{X,\mathbf{n}_t}(\bar{\mathbf{Y}}_{X,s,t})} \tag{191}$$

where X is sampled from $\{V, U, C\}$ with uniform probability $p_X = \frac{1}{3}$ and $\bar{\mathbf{Y}}_{X,s,t}$ is the light subpath resampled from the discrete distribution $p_{X,\mathbf{n}_t}$. Since we use multiple discrete distributions per eye vertex, we must weight the samples using MIS, which is expressed by the factor $w_{X,\mathbf{n}_t}(\bar{\mathbf{Y}}_{X,s,t})$ weighting the contribution.

In our implementation, we use the max heuristic [VG95], which assigns a weight of one to a light subpath only if it is resampled with the distribution giving it the highest probability, and zero otherwise. We optimize our distributions by pre-multiplying every light vertex

CONFIG.	NUMBER OF NODES	PER ITERATION
Door	74	9
Sponza	666	45
Sibenik	1420	85

Table 5 – Number of nodes used per rendering iteration for $\alpha_{accept} = 1$.

probability by its weight and we re-normalize the distributions according to the new probabilities. Thus, we obtain a reduction of variance of the estimator and weights do not need to be evaluated anymore during eye subpath sampling.

11.4 REDUCING THE NUMBER OF DISTRIBUTIONS.

A given view configuration does not require the usage of all nodes of the skeleton. Indeed, many nodes can be located in a part of the scene that is not reachable by eye subpaths. The importance of a node $\mathbf{n}_i$ can be described by the number $N_{mapped}(\mathbf{n}_i)$ of eye vertices mapped to it during the rendering simulation. When this value is small compared to the mean $\bar{N}_{mapped}$ over all nodes, computing the distributions for the node $\mathbf{n}_i$ is not worth the pre-computation time. On our test scenes, more than 50% of the nodes map to less than 1% of the eye vertices.

Based on this observation, we introduce the node acceptance parameter $\alpha_{accept} \in [0,1]$ that is used to ignore nodes that are not mapped to enough eye vertices. The values $\left(N_{mapped}(\mathbf{n}_i)\right)_{i=1...N}$ are accumulated at each iteration and for each node. Before pre-computing the resampling distributions for a given iteration, all nodes $\mathbf{n}_i$ such that $N_{mapped}(\mathbf{n}_i) \leq \alpha_{accept}.\bar{N}_{mapped}$ are discarded for this iteration. We fall back to a uniform resampling distribution for eye vertices that are mapped to these nodes. This optimization increases significantly pre-computation time while keeping the same rendering quality. For our results, we simply set $\alpha_{accept} = 1$, so every node mapping to fewer eye vertices than the mean among all nodes is discarded for the iteration. Table 5 records the number of nodes used per iteration compared to the number of nodes of the filtered skeleton.

11.5 RESULTS

All of our results were rendered on a PC with two CPUs Intel Xeon E5-2650 hexa-core at 2.0 Ghz.

RENDERING CONFIGURATIONS. We compare our algorithm (SkelBPT) against standard BPT and importance caching [GKPS12] adapted to BPT (ICBPT) on three scenes with different view points, lighting configurations and materials. All algorithms sample complete paths having a maximal length of 7 and trace one eye subpath per pixel per iteration. We use the

balance heuristic to compute multiple importance sampling weights for BPT path weights. The max heuristic is used for combining resampling strategies for both our method and ICBPT. The images have a resolution of 1024 x 512 pixels but were cropped to fit in the article. We sample a number of light subpaths equal to the number of pixels at each iteration but we perform connection resampling only on a subset of $N = 1024$ light subpaths randomly chosen in order to keep the overhead of resampling reasonable (both for SkelBPT and ICBPT). We do not apply resampling on paths sampled with the camera projection strategy ($t = 1$) since this strategy requires many light subpaths to be effective. Our implementation of the balance heuristic takes into account the number of paths sampled by each strategy. Our reference images were computed using standard BPT.

ICBPT. We adapted importance caching [GKPS12] to BPT in order to compare our algorithm to a recent similar method. At each iteration, we sample a sparse set of eye subpaths and we use their vertices as importance records (IR) for resampling. The number of IR is controlled by a density parameter $d \in [0, 1]$ such that $d\times$ the number of pixels is the number of eye subpaths traced to position importance records. We set $d = 0.001$ for the presented results, resulting in 524 paths for about 3000 importance records. We use a Kd-tree to store and access to the nearest IRs at each eye vertex.

COMPARISONS. Figures 98, 99 and 100 illustrate visual comparisons of the results produced with our method against standard BPT and ICBPT. For these figures, the rendering time is 300 seconds and the number of iterations performed by each method is indicated as well as the L_1 error (MAE). Both SkelBPT and ICBPT increase rendering quality after a fix number of rendering iterations, as demonstrated by Figure 105. However, ICBPT has a higher cost than our method and the quality gain is not enough to compensate the slower rendering time. Indeed, pre-computing the high number of resampling distributions of ICBPT and accessing them through the Kd-tree is too expensive and the results are worse than expected. Figure 103 shows the convergence curves for the L_1 error. We observe that our method gives a lower error than BPT except for the sibenik scene for which the curves overlap. Table 6 records the time required to reach a given error and the speedup relative to BPT.

In addition to our original publication [NB15], we provide new results in Figures 101 and 101 on two scenes recently acquired. These two scenes presents many occlusions and favor ICBPT over BPT, but our SkelBPT still gives better results than ICBPT. The difference is however less visible than for our three other scenes, especially for the Plants scene for which the error curves almost overlap (Figure 104).

The **Door** scene (Figure 98) features difficult visibility due to the narrow opening of the door. In that case, many light vertices are located behind the door and are connected by BPT with eye vertices of the main room, resulting in many null contributions. Figure 106 illustrates the individual contribution of each of our resampling distributions for this configuration. The scene is provided by Miika Aittala, Samuli Laine, and Jaakko Lehtinen.

The **Sponza** scene (Figure 99) is illuminated by a strong directional light source and lighting is only indirect in this configuration. Our strategy reduces significantly the noise generated by standard BPT by choosing light vertices that are likely to contribute to the final image.

CONFIGURATION	L_1 ERROR	METHOD	TIME (SEC)	SPEEDUP
Door	0.01	BPT	3677.44	× 1
		ICBPT	7095.64	× 0.51
		SkelBPT	2341.36	× 1.57
Sponza	0.15	BPT	1910.39	× 1
		ICBPT	4731.15	× 0.40
		SkelBPT	675.98	× 2.82
Sibenik	0.025	BPT	1334.48	× 1
		ICBPT	3114.65	× 0.42
		SkelBPT	1298.69	× 1.02

Table 6 – Rendering time required to achieve a given error value and speed up relative to BPT.

Despite significant noise reduction on the ground performed by ICBPT, variance is still high as demonstrated by bright spots, especially on the ceiling and draperies. Our reference image has been rendered for 85 hours and still exposes small bright spots on the ground.

The **Sibenik** scene (Figure 100) has less occlusions than the previous ones. The scene is lit by two small area light sources located in corners of the scene. In that configuration, our strategy produces similar results compared to BPT, but additionally slightly reducing visible noise on some parts of the image.

The **Apartment** scene (Figure 101) features a light source located far from the view point (the lighting configuration is the same than the one from Chapter 9). ICBPT performs better than BPT in this configuration, but the result is slightly dark compared to the reference. Our SkelBPT provides a better match, as demonstrated both visually and statistically, from the L1 error.

The **Plants** scene (Figure 102) contains many occlusions, but a simpler lighting configuration (small light bulbs, but located a several places of the scene and close to visible points). In this configuration, ICBPT and SkelBPT renders similar results and the convergence curves almost overlap. This can be explained by the simpler lighting configuration and the fact that the scene is not too large. A good density of importance records can be obtained for ICBPT in that configuration and thus the algorithm gives good performance.

11.6 CONCLUSION AND PERSPECTIVES

We presented a new method to improve the efficiency of bidirectional path tracing by using a skeleton of the empty parts of the scene. We demonstrated experimentally that taking advantage of this skeleton leads to a simple and efficient resampling strategy for algorithms based on vertex connection. We discuss here some limitations and future works.

COMBINING OUR METHOD WITH ICBPT. Importance caching is extremely efficient when applied to many-light rendering. Our first experiments were actually performed on MLR and demonstrated that IC outperforms our method on this algorithm. The reason for this is that MLR only connects points visible from the camera to light vertices (VPLs). This set of visible points can be covered with a limited number of importance records for achieving precise resampling. For BPT, it would be interesting to use IC at eye vertices visible from the camera and our skeleton based resampling at remaining eye vertices. The method would then benefit from a costly but robust resampling for the directly visible eye vertex of each path and a cheaper but coarse resampling for others.

GPU IMPLEMENTATION. Our method can take advantage of a GPU implementation since our skeleton is static and sparse. A shadow map can be pre-computed for each node and used to compute our resampling distributions that require many visibility tests. Using this strategy would be more efficient than tracing shadow rays since the visibility test could thus be performed in constant time.

COMPARISON WITH "PROBABILISTIC CONNECTIONS FOR BIDIRECTIONAL PATH TRACING" A future work is to compare our method to the recent technique addressing the same problem, introduced by Popov et al. [PRDD15]. This method follows the same idea than Importance Caching, but provides a new MIS strategy to handle correlated paths with better efficiency.

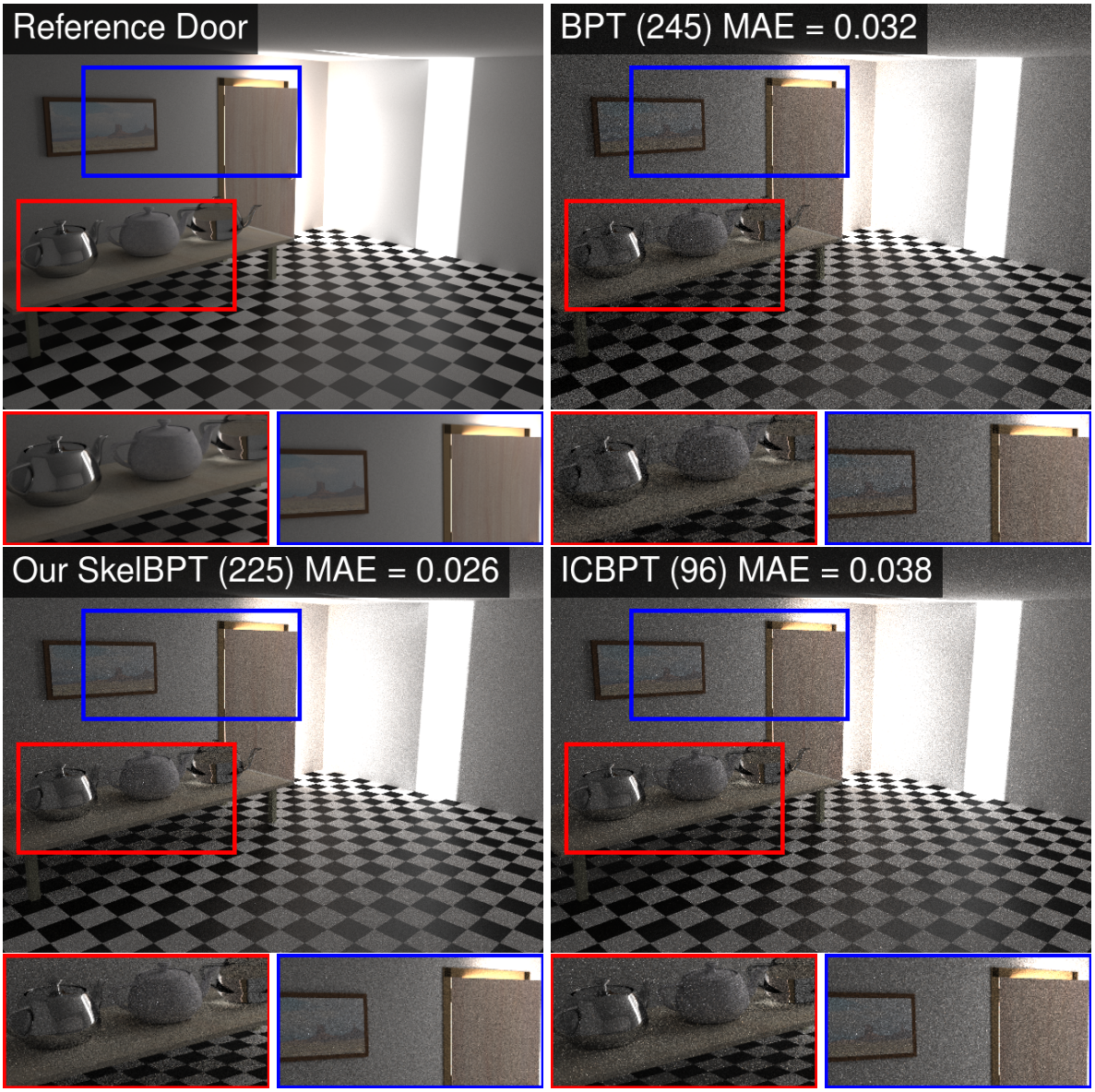


Figure 98 – Visual comparison of BPT, SkelBPT and ICBPT on the Door scene for a rendering time of 300 seconds.

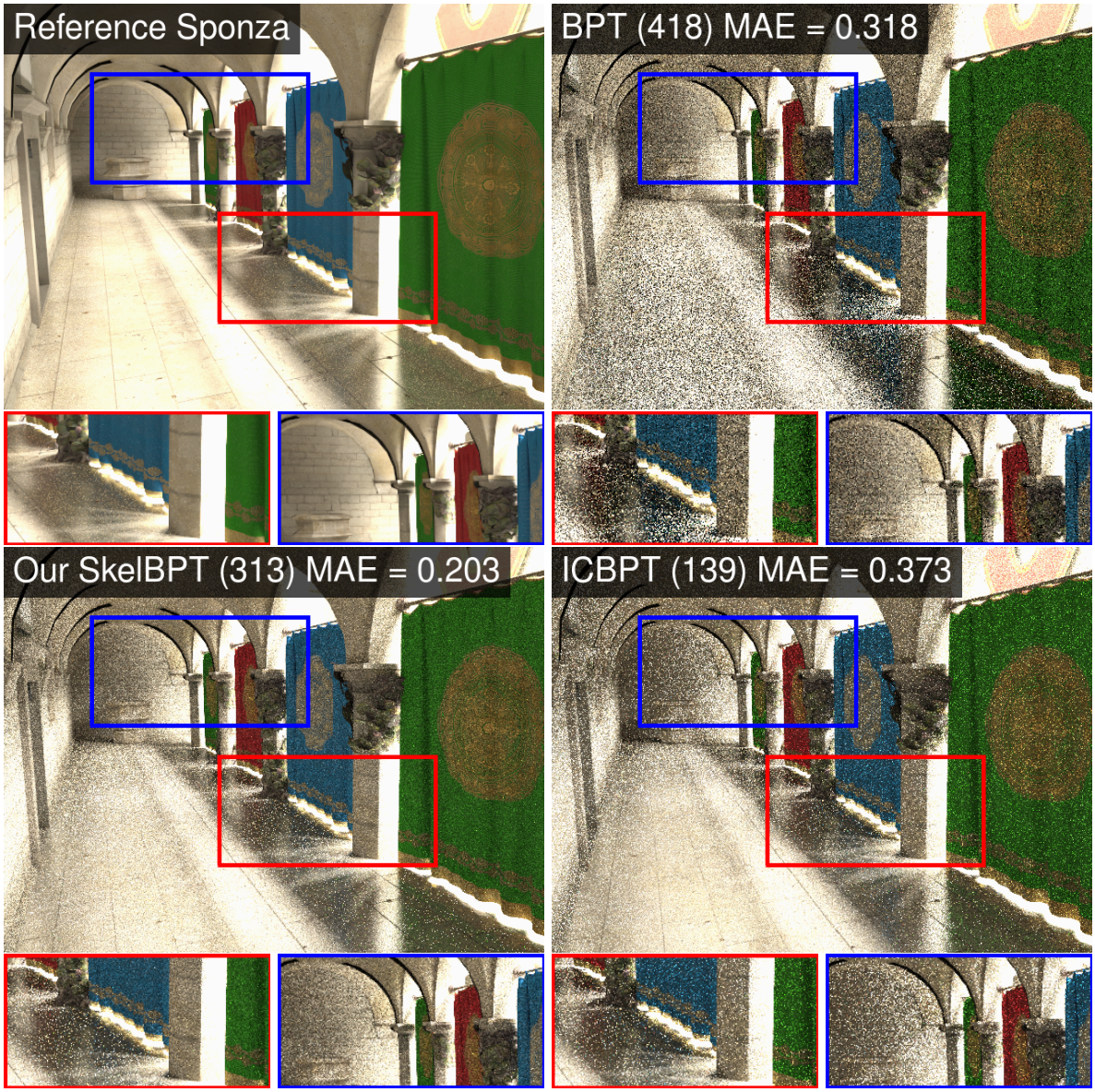


Figure 99 – Visual comparison of BPT, SkelBPT and ICBPT on the Sponza scene for a rendering time of 300 seconds.

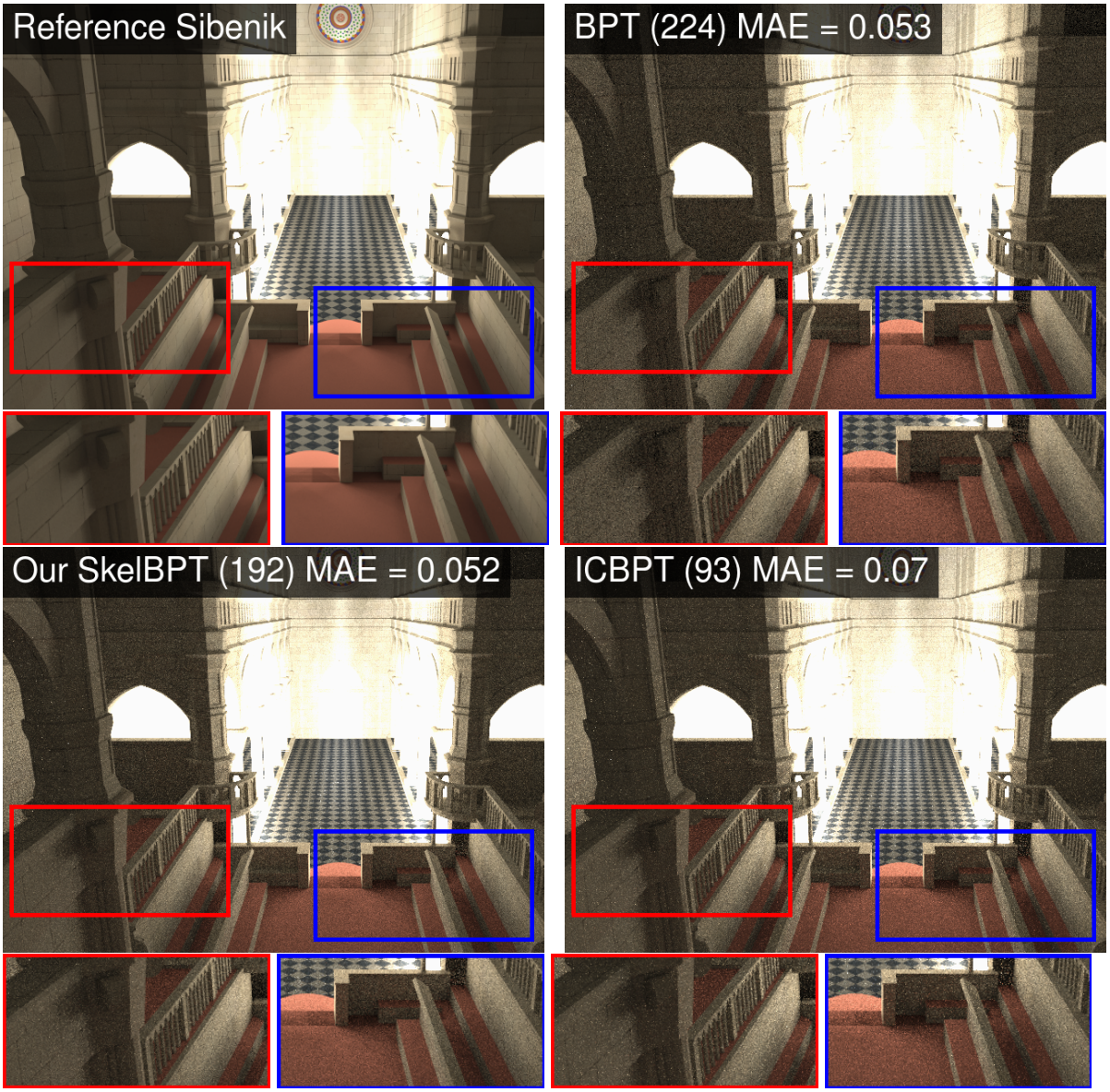


Figure 100 – Visual comparison of BPT, SkelBPT and ICBPT on the Sibenik scene for a rendering time of 300 seconds.

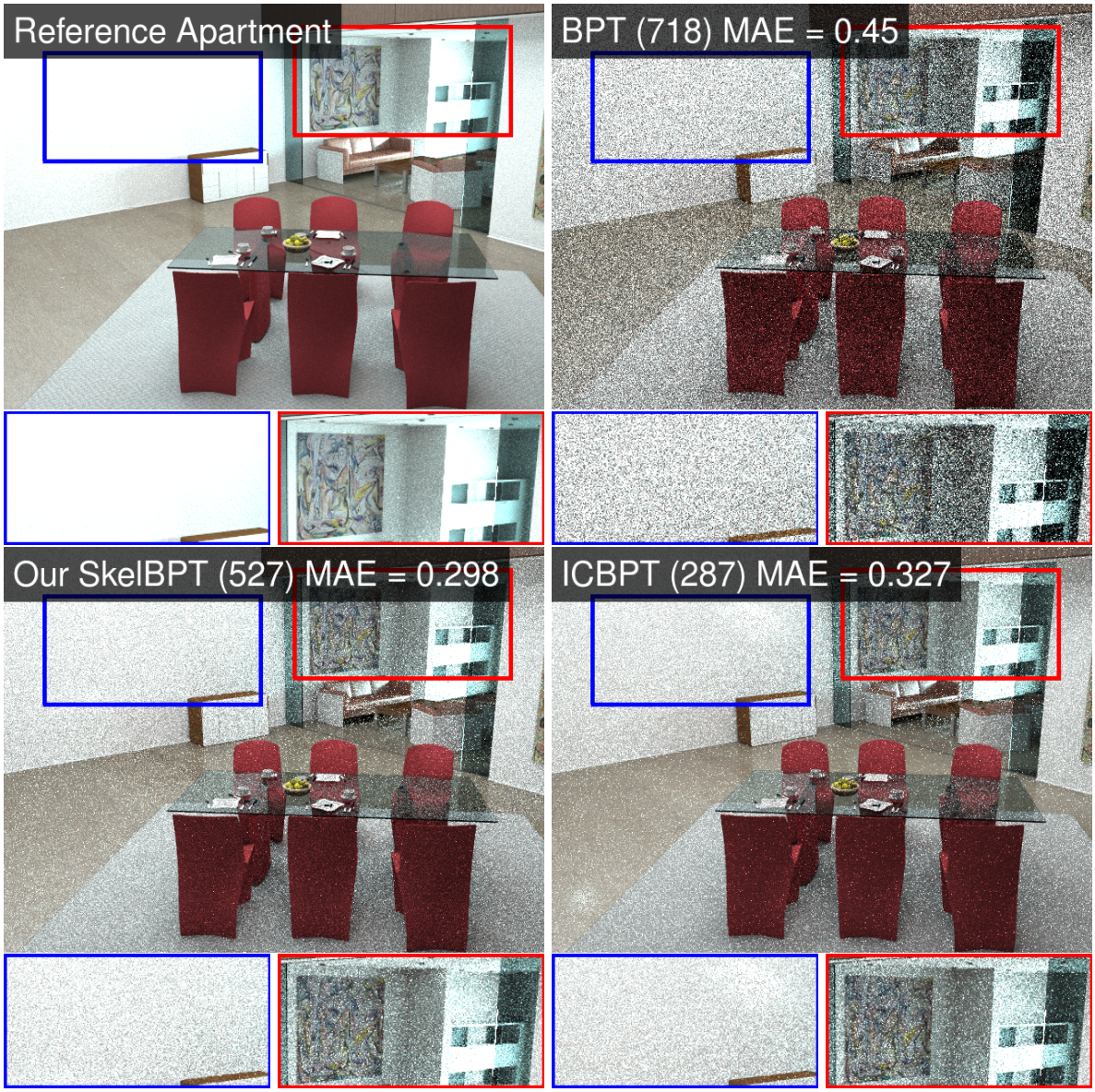


Figure 101 – Visual comparison of BPT, SkelBPT and ICBPT on the Apartment scene for a rendering time of 900 seconds.

Figure 102 – Visual comparison of BPT, SkelBPT and ICBPT on the Plants scene for a rendering time of 900 seconds.

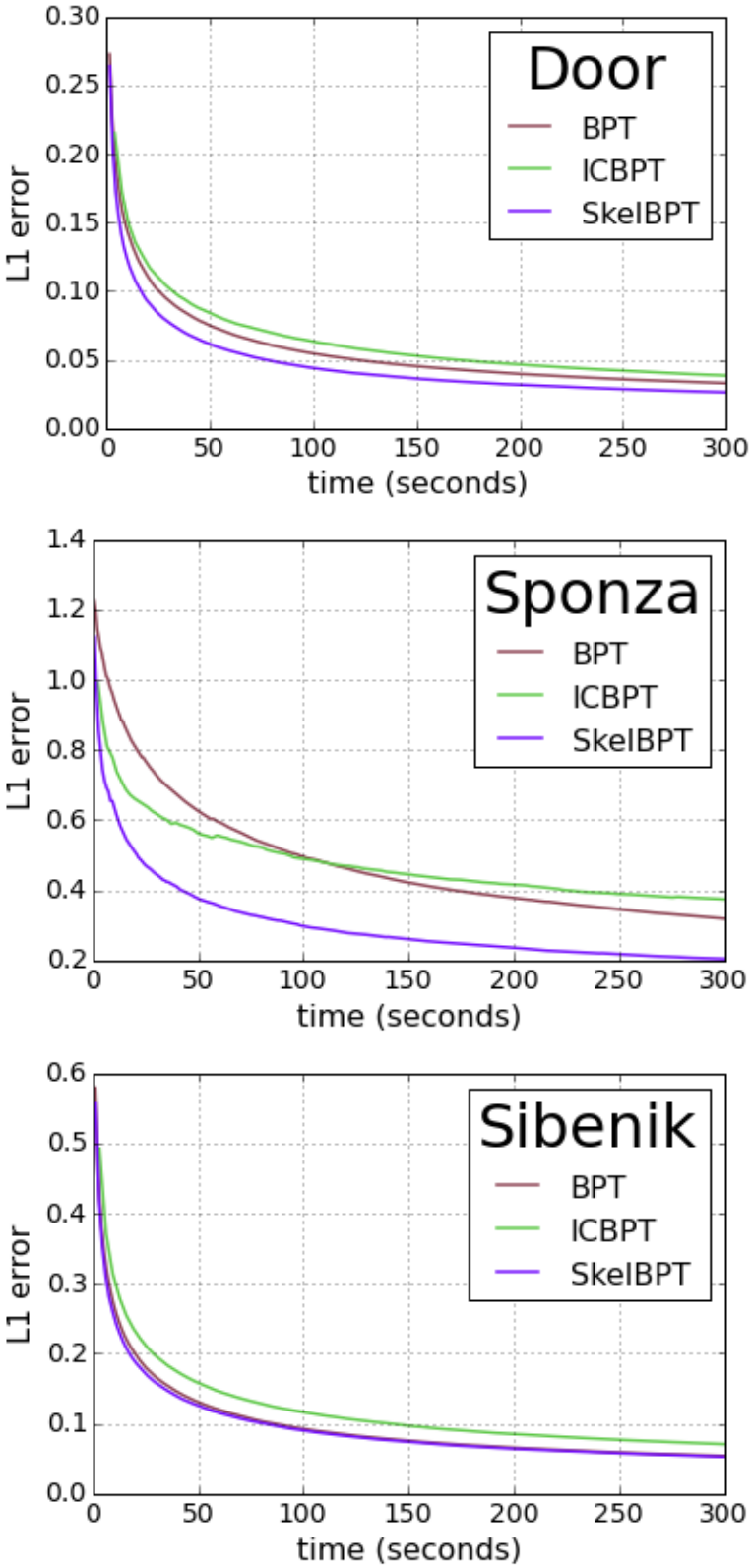


Figure 103 – L1-error curves for a rendering time of 300 seconds.

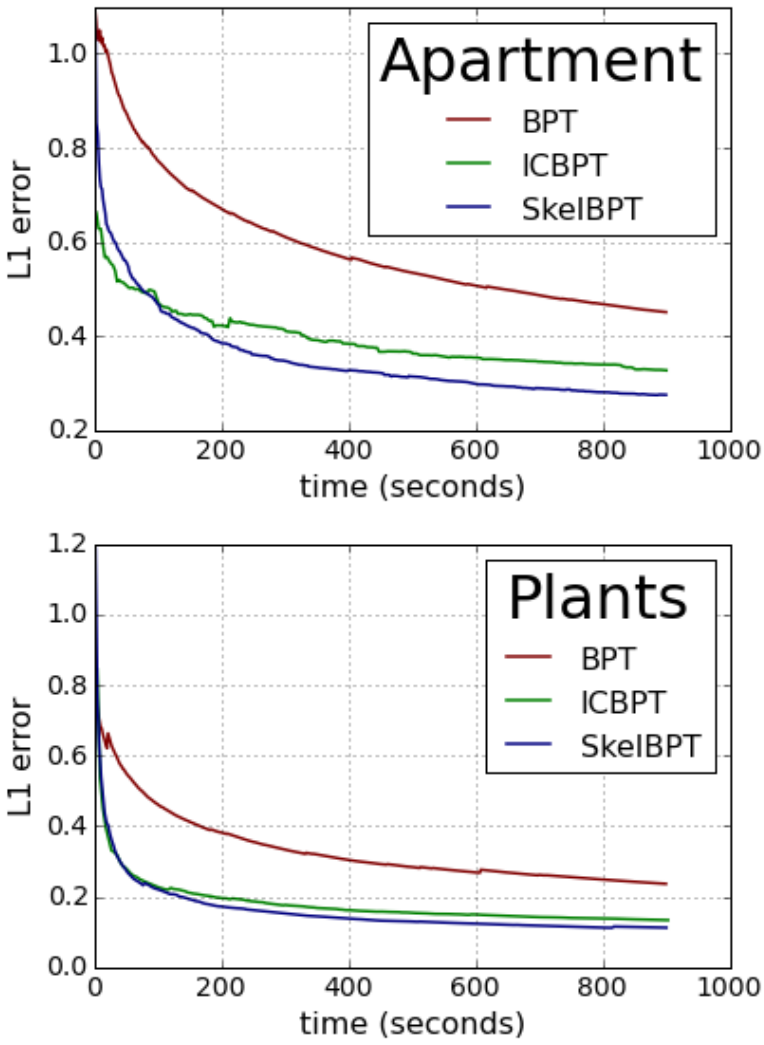


Figure 104 – L1-error curves for a rendering time of 900 seconds.

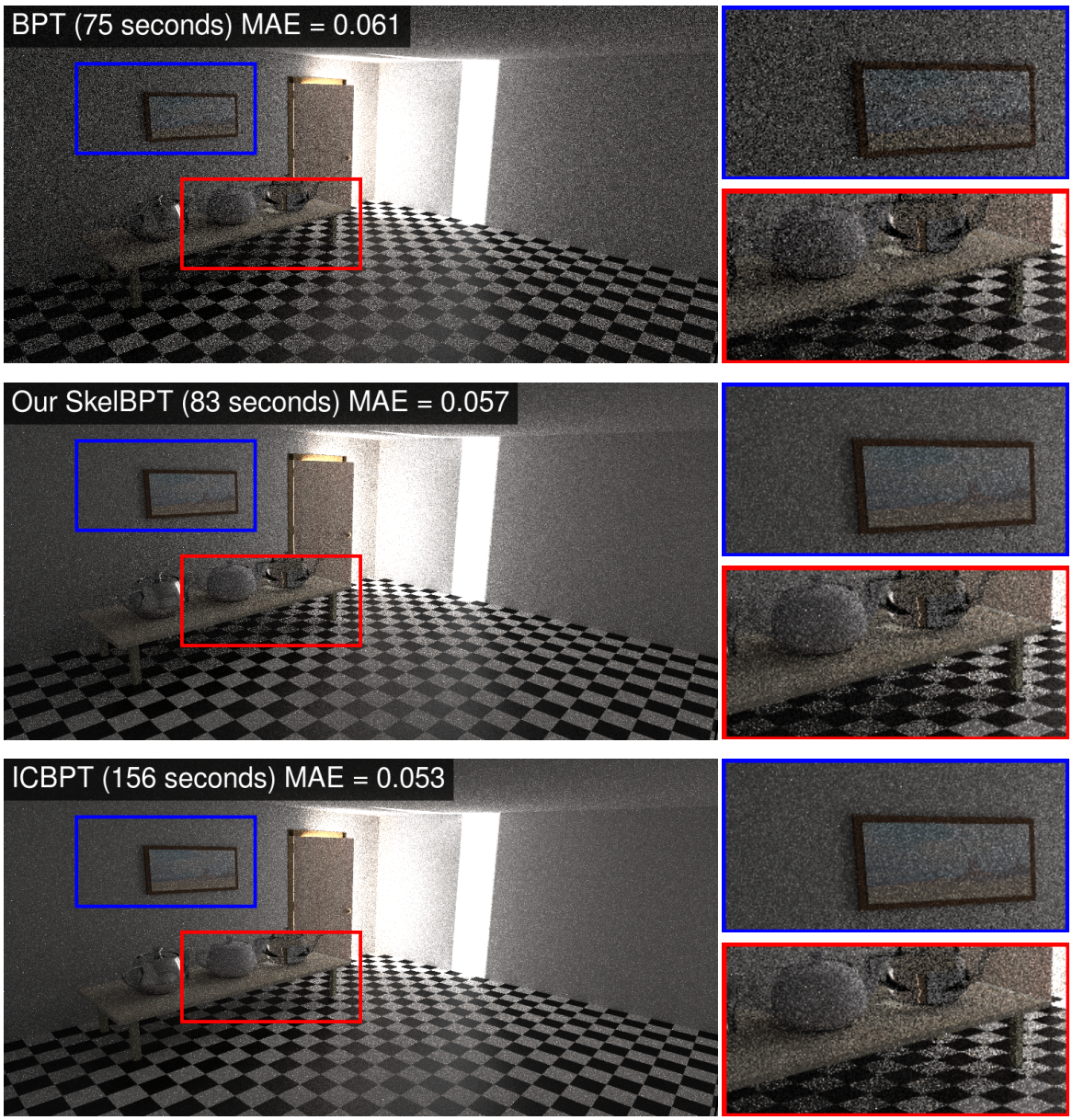


Figure 105 – Comparison of the three methods after 64 rendering iterations. Given the same number of rendering iterations, ICBPT gives a better result on this configuration. However, the time required by ICBPT is approximately twice the rendering time of BPT. The quality gain is not enough to compensate the overhead introduced by ICBPT, as opposed to our SkelBPT.

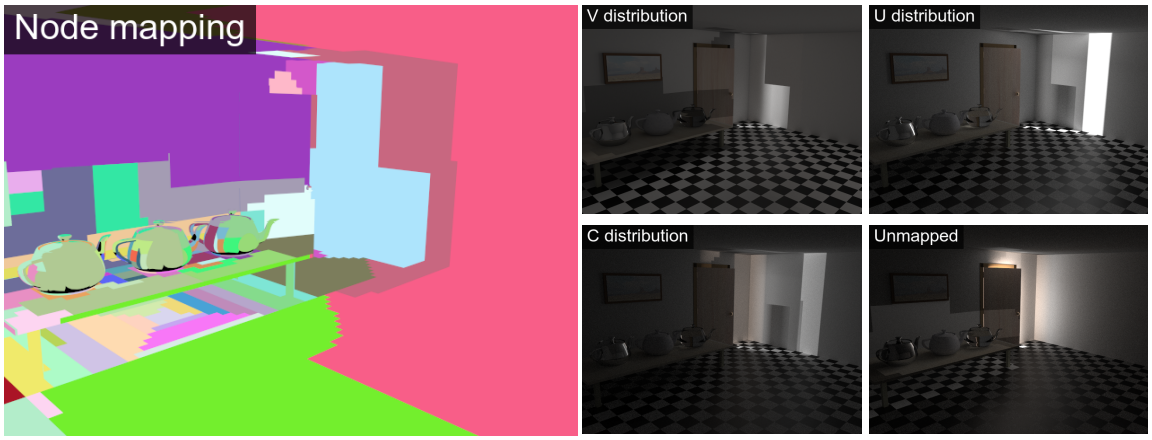


Figure 106 – Individual contribution of each of our distributions on the Door scene after two hours of rendering. Leftmost image illustrates the segmentation of the scene according to our skeleton node mapping. Rightmost image is the contribution associated to the default uniform resampling distribution, used when the node mapped to a surface point has been discarded by our optimization at a given iteration. Multiple importance sampling blends the contributions such that the discontinuities produced by node mapping are not visible in the final result.

12

CONCLUSION AND PERSPECTIVES

This thesis presented new contributions in the field of physically based rendering, with the particularity of using tools from the discrete shape analysis framework to improve the efficiency of existing rendering approaches. Despite the relationship between rendering and geometric/topological configurations, few existing works have effectively considered using a discrete skeleton, or distance-based maps to address rendering problems. Our works constitute first attempts to take advantage of these features, and we demonstrated experimentally that they can bring useful information about visibility in order to address the rendering of highly occluded scenes.

12.1 SUMMARY OF CONTRIBUTIONS

In the introduction, we stated the following questions that have driven our works, and we believe our contributions helped to provide answers:

12.1.1 *How can we use a topological skeleton to make rendering algorithms based on path sampling more robust in occluded scenes ?*

Occluded scenes, lit by strong light sources, generally present highly indirect illumination at each surface point. Not taking this illumination into account for path sampling lead to inefficient sampling strategies, that do not distribute paths according to their contribution and thus induce low convergence rate for rendering algorithms. Existing approaches to address this problem are mostly based on particle tracing and density estimation to reconstruct pdfs that mimic incident illumination. These methods are often costly in terms of memory, and their efficiency depend on the quality of the particle distribution. While progressive strategies can help to offset these issues, the analysis of the geometry and topology of the scene remained ignored, loosing the opportunity of using additional information to drive ray sampling or vertex connection. Our works presented in Chapters 9 and Chapters 11 take advantage of a topological curvilinear skeleton of the empty space of the scene to improve rendering algorithms based on path sampling. This skeleton offers and sparse representation of the geometry and topology of the empty space, which can be efficiently manipulated to pre-compute information on the main global light flow, to guide ray sampling, and on the local visibility at different regions of the empty space, to improve vertex connections.

12.1.2 *Is it possible to take advantage of the same kind of skeleton to also improve real-time rendering of global illumination ?*

We took the decision to use the skeleton for fast approximation of indirect illumination for real-time rendering algorithms, based on VPLs. Real-time rendering algorithms are more sensitive to bad decisions that can be made at some surface points, due to some false or incomplete information stored at the skeleton. The method presented in Chapter 7 directly suffers from these decisions, by demonstrating discontinuity artifacts or bad approximations at some parts of the scene. The lessons learnt from the development of this method led us to our "Skeleton Shadow Mapping" method, presented in Chapter 8, which is based on more conventional rendering methods and use the skeleton as a coarse visibility proxy, to alleviate the costly computation of shadow maps for many virtual light sources. While this work is still in progress, our results are promising regarding the real-time processing of thousands of VPLs, which is required for smooth rendering of highly directional materials.

12.1.3 *How to take advantage of the information stored by an opening map to guide path sampling on regions of empty space that are usually hard to explore ?*

An opening map describes, at each 3D point, the local thickness of the object under analysis, the empty space in our case. Narrow regions of the empty space are usually hard to explore using local path sampling and the opening map offers a practical way of identifying these regions and tracing rays directly through them, as demonstrated by our portal extraction method presented in Chapter 10. While these portals effectively give a solution to explore the empty space more efficiently, our application to path tracing suffers from discontinuities because of fast changes between sampling strategies at neighbor surface points. Future works will be devoted to a good use of portals for rendering, with correct combination with standard BSDF sampling, using better weighting functions for multiple importance sampling.

12.2 FUTURE WORKS

Overall, we acknowledge that our works lack comparisons with concurrent recent works. Regarding real-time rendering, since we addressed the problem of indirect shadows approximation, we plan to compare our methods to imperfect shadow maps [RGK+08, REH+11, BBH13] and ManyLODs [HREB11], that constitute the main existing solutions for rendering shadows from a large set of VPLs. For ray sampling according to incident illumination or importance, the state-of-the-art solution is the method from Vorbal et al. [VKŠ+14], so we plan to compare our skeleton ray sampling method and portal based sampling method to it. We compared our vertex connection resampling algorithm based on the skeleton to an adaptation of the importance caching method [GKPS12] to BPT. A recent solution, introduced by Popov et al. [PRDD15], addresses the same problem directly for BPT. Therefore, we want to provide a comparison with our method and eventually combine the two algorithms for better efficiency.

Some of our works are still in progress (Chapter 8, Chapter 10) and will be completed in future works to be submitted for publication.

We limit ourselves to the use of a curvilinear skeleton for our methods based on this tool. Some scenes are not well described by a curvilinear skeleton, because of regions of empty spaces less elongated in one particular direction. The next step is to take advantage of a surface skeleton, in order to ensure a better representation of the dimensionality of the empty space along different axis. We used the opening map alone for the method presented in Chapter 10. Combining it with the skeleton could give further information to develop a robust rendering algorithm, but further research is required regarding that goal.

Finally, our works need more theoretical analysis regarding their efficiency. This analysis is quite hard to develop, since our tools are computed in discrete spaces. However, it could be possible to perform variance and error analysis directly in the discrete framework, and to extend it to the continuous one. We plan to work in that direction for future works, in order to demonstrate formally that our discrete tools are well adapted to light transport simulation.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

PT Path Tracing

BPT Bidirectional Path Tracing

MLR Many-Light Rendering

PM Photon Mapping

PPM Progressive Photon Mapping

MLT Metropolis Light Transport

MCMC Markov Chain Monte Carlo

VPL Virtual Point Light

VSGL Virtual Spherical Gaussian Light

MC Monte Carlo

MIS Multiple Importance Sampling

IR Importance Record

BSDF Bidirectional Scattering Distribution Function

DSA Discrete Shape Analysis

pdf Probability Density Function

cdf Cumulative Distribution Function

pmf Probability Mass Function

BIBLIOGRAPHY

[AK90] James Arvo and David Kirk. Particle transport and image synthesis. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 63–66, 1990.

[AK04] Rosenfeld Azriel and Reinhard Klette. *Digital Geometry: Geometric Methods for Digital Picture Analysis*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2004.

[Ant11] Dietger Van Antwerpen. Recursive mis computation for streaming bdpt on the gpu, 2011.

[ARB90] John M. Airey, John H. Rohlf, and Frederick P. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. *SIGGRAPH Comput. Graph.*, 24(2):41–50, February 1990.

[Arv95] James Richard Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, New Haven, CT, USA, 1995. AAI9619140.

[BApS02] Stefan Brabec, Thomas Annen, and Hans peter Seidel. Shadow mapping for hemispherical and omnidirectional light sources. In *In Proc. of Computer Graphics International*, pages 397–408, 2002.

[BBH13] Tomas Barák, Jiri Bittner, and Vlastimil Havran. Temporally Coherent Adaptive Sampling for Imperfect Shadow Maps. *Computer Graphics Forum*, 2013.

[BC06] Gilles Bertrand and Michel Couprie. A new 3D parallel thinning scheme based on critical kernels. In *Discrete Geometry for Computer Imagery*, pages 580–591. Springer, 2006.

[BC09] Gilles Bertrand and Michel Couprie. On parallel thinning algorithms: Minimal non-simple sets, p-simple points and critical kernels. *Journal of Mathematical Imaging and Vision*, 35(1):23–35, 2009.

[BC12] Venceslas Biri and John Chaussard. Skeleton based importance sampling for path tracing. In *proceedings of Eurographics 2012*, pages 1–4, mar 2012. short papers.

[BC14] Gilles Bertrand and Michel Couprie. Powerful parallel and symmetric 3d thinning schemes based on critical kernels. *Journal of Mathematical Imaging and Vision*, 48(1):134–148, 2014.

[BC15] Gilles Bertrand and Michel Couprie. Isthmus based parallel and symmetric 3d thinning algorithms. *Graphical Models*, 80:1–15, 2015.

[Ber07] Gilles Bertrand. On critical kernels. *Comptes Rendus de l'Académie des Sciences, Série Mathématiques*, I(345):363–367, 2007.

[Blu67] Harry Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Proc. Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, MA, November 1967. MIT Press.

[BM94] Gilles Bertrand and Grégoire Malandain. A new characterization of three-dimensional simple points. *Pattern Recogn. Lett.*, 15(2):169–175, February 1994.

[CB08] Michel Couprie and Gilles Bertrand. New characterizations of simple points, minimal non-simple sets and p-simple points in 2d, 3d and 4d discrete spaces. In *Discrete Geometry for Computer Imagery, 14th IAPR International Conference, DGCI 2008, Lyon, France, April 16-18, 2008. Proceedings*, pages 105–116, 2008.

[CB09] Michel Couprie and Gilles Bertrand. New characterizations of simple points in 2D, 3D and 4D discrete spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):637–648, April 2009.

[CB14] Michel Couprie and Gilles Bertrand. Isthmus-based parallel and asymmetric 3d thinning algorithms. In *Discrete Geometry for Computer Imagery - 18th IAPR International Conference, DGCI 2014, Siena, Italy, September 10-12, 2014. Proceedings*, pages 51–62, 2014.

[CC09] John Chaussard and Michel Couprie. Surface thinning in 3d cubical complexes. In *Proceedings of the 13th International Workshop on Combinatorial Image Analysis*, IWCIA '09, pages 135–148, Berlin, Heidelberg, 2009. Springer-Verlag.

[CCT09] John Chaussard, Michel Couprie, and Hugues Talbot. A discrete lambda-medial axis. In Srecko Brlek, Christophe Reutenauer, and Xavier Provençal, editors, *Discrete Geometry for Computer Imagery*, volume 5810 of *Lecture Notes in Computer Science*, pages 421–433. Springer, 2009.

[CCT10a] John Chaussard, Michel Couprie, and Hugues Talbot. Robust skeletonization using the discrete lambda-medial axis. *Pattern Recognition Letters*, In Press,, 2010.

[CCT10b] John Chaussard, Michel Couprie, and Hugues Talbot. Robust skeletonization using the discrete lambda-medial axis. *Pattern Recognition Letters*, In Press, Corrected Proof:–, 2010.

[CCZ07] Michel Couprie, David Coeurjolly, and Rita Zrour. Discrete bisector function and euclidean skeleton in 2d and 3d. *Image and Vision Computing*, 25(10):1543–1556, 2007.

[Cha10] John Chaussard. *Topological tools for discrete shape analysis*. PhD thesis, Université Paris-Est, December 2010.

[CNBC13] John Chaussard, Laurent Noël, Venceslas Biri, and Michel Couprie. A 3d curvilinear skeletonization algorithm with application to path tracing. In *Discrete Geometry for Computer Imagery*, volume 7749 of *Lecture Notes in Computer Science*, pages 119–130. Springer Berlin Heidelberg, 2013.

[CNS+11] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, volume 30, pages 1921–1930. Wiley Online Library, 2011.

[COCSD03] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431, July 2003.

[Coe12] David Coeurjolly. Fast and accurate approximation of digital shape thickness distribution in arbitrary dimension. *Comput. Vis. Image Underst.*, 116(12):1159–1167, December 2012.

[Cou11] Michel Couprie. Hierarchic euclidean skeletons in cubical complexes. In *Discrete Geometry for Computer Imagery - 16th IAPR International Conference, DGCI 2011, Nancy, France, April 6-8, 2011. Proceedings*, pages 141–152, 2011.

[Dev86] Luc Devroye. Non-uniform random variate generation, 1986.

[DGR+09] Zhao Dong, Thorsten Grosch, Tobias Ritschel, Jan Kautz, and Hans-Peter Seidel. Real-time indirect illumination with clustered visibility. In *VMV*, pages 187–196, 2009.

[Dij71] E.W. Dijkstra. *EWD316: A Short Introduction to the Art of Programming*. Technische Hogeschool, 1971.

[DKH+14] Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104, 2014.

[DLW93] Philip Dutré, Eric P. Lafortune, and Yves D. Willems. Monte carlo light tracing with direct computation of pixel intensities. In *3rd International Conference on Computational Graphics and Visualisation Techniques*, pages 128–137, Alvor, Portugal, December 1993.

[DS05] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 203–231, New York, NY, USA, 2005. ACM.

[FMH05] David Fradin, Daniel Meneveaux, and Sebastien Horna. Out of core photon-mapping for large buildings. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, pages 65–72, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[GKDS12] Iliyan Georgiev, Jaroslav Krivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31:192:1–192:10, November 2012.

[GKPS12] Iliyan Georgiev, Jaroslav Křivánek, Stefan Popov, and Philipp Slusallek. Importance caching for complex illumination. In *Computer Graphics Forum*, volume 31, pages 701–710, 2012.

[GS10] Iliyan Georgiev and Philipp Slusallek. Simple and robust iterative importance sampling of virtual point lights. In H. P. A. Lensch and S. Seipel, editors, *Proceedings of Eurographics 2010 (short papers)*, pages 57–60, Norrkoping, Sweden, 2010. Eurographics Association.

[Hei14] Eric Heitz. Understanding the masking-shadowing function in microfacet-based brdfs. *Journal of Computer Graphics Techniques (JCGT)*, 3(2):48–107, June 2014.

[Hil69] C. J. Hilditch. Linear skeletons from square cupboards. In *Machine Intelligence*, 1969.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual spherical lights for many-light rendering of glossy scenes. In *ACM Transactions on Graphics (TOG)*, volume 28, page 143. ACM, 2009.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, volume 27, page 130, 2008.

[HP02] Heinrich Hey and Peter Purgathofer. Importance sampling with hemispherical particle footprints. In *Spring Conference on Computer Graphics 2002*, pages

[HPB07] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.*, 26(3), July 2007.

[HPJ12] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)*, 31(6):191, 2012.

[HR08] Wim H. Hesselink and Jos B. T. M. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2204–2217, 2008.

[HREB11] Matthias Hollander, Tobias Ritschel, Elmar Eisemann, and Tamy Boubekeur. Manylods: Parallel many-view level-of-detail selection for real-time global illumination. In *Computer Graphics Forum*, volume 30, pages 1233–1240. Wiley Online Library, 2011.

[Jak10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[Jen95] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Eurographics Rendering Workshop*, pages 326–335, 1995.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. Natick, MA, USA, 2001.

[Kaj86] James T. Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150, 1986.

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, 1997.

[KFF+15] A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols. The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, pages 24:1–24:7, New York, NY, USA, 2015. ACM.

[KH01] Alexander Keller and Wolfgang Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276, London, UK, UK, 2001. Springer-Verlag.

[KK04] Thomas Kollig and Alexander Keller. Illumination in the Presence of Weak Singularities. In *MCQMC Methods*, 2004.

[KMW91] Ralph Kopperman, PaulR. Meyer, and RichardG. Wilson. A jordan surface theorem for three-dimensional digital spaces. *Discrete & Computational Geometry*, 6(2):155–161, 1991.

[Kon97] T. Yung Kong. Topology-preserving deletion of 1's from 2-, 3- and 4-dimensional binary images. In *Proceedings of the 7th International Workshop on Discrete Geometry for Computer Imagery*, DGCI '97, pages 3–18, London, UK, UK, 1997. Springer-Verlag.

[KSK01] Csaba Kelemen and L'aszl'o Szirmay-Kalos. Simple and robust mutation strategy for metropolis light transport algorithm. Technical Report TR-186-2-01-18, July 2001.

[LCLJ10] Lu Liu, Erin Wolf Chambers, David Letscher, and Tao Ju. A simple and robust thinning algorithm on cell complexes. In *Computer Graphics Forum (Proceedings of Pacific Graphics 2010)*, 2010.

[Lie03] André Lieutier. Any open bounded subset of rn has the same homotopy type than its medial axis. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, pages 65–75, New York, NY, USA, 2003. ACM.

[Loh01] Christophe Lohou. *Contribution à l'analyse topologique des images: étude d'algorithmes de squelettisation pour images 2D et 3D, selon une approche topologie digitale ou topologie discrète.* PhD thesis, Université de Marne-La-Vallée, 2001.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of the 3rd international conference on computational graphics and visualization techniques*, pages 145–153, 1993.

[LW94] Eric P. Lafortune and Yves D. Willems. Using the modified phong reflectance model for physically based rendering. Technical report, 1994.

[LW95] Eric P Lafortune and Yves D. Willems. Reducing the number of shadow rays in bidirectional path tracing. In *Proceedings of The Winter School of Computer Graphics and Visualisation '95*, volume 95, pages 384–392, February 1995.

[Mat67] Georges Matheron. *Eléments pour une Théorie des Milieux Poreux*. 1967.

[MHA14] Maxime Maria, Sébastien Horna, and Lilian Aveneau. Topological Space Partition for Fast Ray Tracing in Architectural Models. In *GRAPP 2014 - 9th International Joint Conference on Computer Graphics Theory and Applications*, pages 225 – 235, Lisbon, Portugal, January 2014.

[ML00] Rémy Malgouyres and Alexandre Lenoir. Topology preservation within digital surfaces. *Graph. Models*, 62(2):71–84, March 2000.

[MMB98] D. Meneveaux, E. Maisel, and K. Bouatouch. A new partitioning method for architectural environments. Novembre 1998. Journal of Vizualisation and Computer Animation, Volume 9 (1998), Issue 4, Wiley Publishers, pp 195-213.

[MPBM03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[MU49] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical Association*, 44(247):335–341, 1949.

[NB15] Laurent Noël and Venceslas Biri. Skeleton based vertex connection resampling for bidirectional path tracing. In *23rd Pacific Conference on Computer Graphics and Applications*, Pacific Graphics, 2015.

[Noë15] Laurent Noël. Skeleton based vertex connection resampling for bidirectional path tracing - code, 2015. https://github.com/Celeborn2BeAlive/pg2015-code.

[OBA12] Ola Olsson, Markus Billeter, and Ulf Assarsson. Clustered deferred and forward shading. In *HPG '12: Proceedings of the Conference on High Performance Graphics 2012*, 2012.

[OP11] Jiawei Ou and Fabio Pellacini. Lightslice: Matrix slice sampling for the many-lights problem. *ACM Trans. Graph.*, 30(6):179:1–179:8, December 2011.

[OPB15] Ola Olsson, Emil Persson, and Markus Billeter. Real-time many-light management and shadows with clustered shading. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, pages 12:1–12:398, New York, NY, USA, 2015. ACM.

[OSK+14] Ola Olsson, Erik Sintorn, Viktor Kämpe, Markus Billeter, and Ulf Assarsson. Efficient virtual shadow maps for many lights. In *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 87–96. ACM, 2014.

[Pal07] Kálmán Palágyi. A Subiteration-Based Surface-Thinning Algorithm with a Period of Three. In *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 294–303. Springer Berlin / Heidelberg, 2007.

[PBPP11] Anthony Pajot, Loïc Barthe, Mathias Paulin, and Pierre Poulin. Combinatorial bidirectional path-tracing for efficient hybrid cpu/gpu rendering. *Computer Graphics Forum*, 30(2):315–324, 2011.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[PK98] Kálmán Palágyi and Attila Kuba. A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recogn. Lett.*, 19(7):613–627, May 1998.

[PPI98] Ingmar Peter, Georg Pietrek, and Fachbereich Informatik. Importance driven construction of photon maps. In *In Rendering Techniques '98 (Proceedings of the 9th Eurographics Workshop on Rendering*, pages 269–280. Springer-Verlag, 1998.

[PRDD15] Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. Probabilistic connections for bidirectional path tracing. In *Computer Graphics Forum*, volume 34, page 12, 2015.

[RC11] Benjamin Raynal and Michel Couprie. Isthmus-based 6-directional parallel thinning algorithms. In *Discrete Geometry for Computer Imagery - 16th IAPR International Conference, DGCI 2011, Nancy, France, April 6-8, 2011. Proceedings*, pages 175–186, 2011.

[REH+11] Tobias Ritschel, Elmar Eisemann, Inwoo Ha, James DK Kim, and Hans-Peter Seidel. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. In *Computer Graphics Forum*, volume 30, pages 2258–2269. Wiley Online Library, 2011.

[RGK+08] Tobias Ritschel, Thorsten Grosch, Min H. Kim, H.-P. Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect shadow maps for efficient computation of indirect illumination. 27(5):129, 2008.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph.*, 21(4):283–291, August 1987.

[SHD15] Florian Simon, Johannes Hanika, and Carsten Dachsbacher. Rich-vpls for improving the versatility of many-light methods. *Computer Graphics Forum (Proceedings of Eurographics)*, 34(2):575–584, May 2015.

[SIMP06a] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, pages 389–397, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[SIMP06b] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '06, pages 53–60, New York, NY, USA, 2006. ACM.

[SIP07] B. Segovia, J.C. Iehl, and B. Péroche. Metropolis instant radiosity. *Computer Graphics Forum*, 26(3):425–434, 2007.

[SL06] J. Steinhurst and A. Lastra. Global Importance Sampling of Glossy Surfaces Using the Photon Map. *Symposium on Interactive Ray Tracing*, 0:133–138, 2006.

[SS10] Michael Schwarz and Hans-Peter Seidel. Fast parallel surface and solid voxelization on gpus. *ACM Trans. Graph.*, 29(6):179:1–179:10, December 2010.

[SSB15] Punam Saha, Robin Strand, and Gunilla Borgefors. Digital topology and geometry in medical imaging: a survey. 2015.

[TCE05] Justin F. Talbot, David Cline, and Parris Egbert. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics conference on Rendering Techniques*, pages 139–146, 2005.

[Tel92] Seth Jared Teller. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, Berkeley, CA, USA, 1992. UMI Order No. GAX93-30757.

[TFFH94] Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity computations. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 443–450, New York, NY, USA, 1994. ACM.

[Tok14] Yusuke Tokuyoshi. Virtual spherical gaussian lights for real-time glossy indirect illumination. In *SIGGRAPH Asia 2014 Technical Briefs*, SA '14, pages 17:1–17:4, New York, NY, USA, 2014. ACM.

[TS91] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph.*, 25(4):61–70, July 1991.

[Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, 1997.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, 1995.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, 1997.

[Vin94] Luc Vincent. Fast opening functions and morphological granulometries. In *Proc. SPIE 2300, Image Algebra and Morphological Image Processing V*, volume 2300, pages 253–267, July 1994.

[VKŠ+14] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. Online learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics*, 33(4):1–11, July 2014.

[WABG06] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1081–1088, 2006.

[WFA+05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1098–1107. ACM, 2005.