

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

François DESRICHARD

Le 6 décembre 2021

Analyse de l'espace des chemins pour la composition des ombres et lumières

---

École doctorale : **EDMITT - École Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
David VANDERHAEGHE

Jury

M. Adrien BOUSSEAU, Rapporteur
Directeur de recherche, Inria Sophia-Antipolis

M. Pascal BARLA, Rapporteur
Chargé de recherche, Inria Bordeaux Sud-Ouest

Mme Joëlle THOLLOT, Examinatrice
Professeure, Université Grenoble Alpes

M. Mathias PAULIN, Examinateur
Professeur, Université Toulouse 3 - Paul Sabatier

M. Pierre POULIN, Examinateur
Professeur, Université de Montréal

M. David VANDERHAEGHE, Directeur de thèse
Maître de conférences, Université Toulouse 3 - Paul Sabatier

Remerciements

Les trois années de thèse qui viennent de filer derrière moi auront été inoubliables, et aussi agréables qu'intenses. C'est à de nombreuses personnes que je dois cette belle expérience, et je souhaite prendre un moment pour les remercier.

Je salue en premier lieu mon encadrant David, qui a su guider mon parcours avec sensibilité, tact et clairvoyance. Notre relation aura été complémentaire quand il s'agissait de recherche, et profondément complice sur tout le reste. Merci à Mathias d'avoir toujours gardé la porte ouverte pour me recevoir et discuter passionnément d'informatique, de physique, de ski et de la vie en général. Je n'oublierai pas Loïc et Nicolas, qui sont capables en un trait d'humour de gommer les situations difficiles. Une pensée aussi pour Tamy, qui m'a transmis le goût de l'informatique graphique et m'a permis de rejoindre Toulouse.

Merci au jury : Adrien, Pascal, Joëlle et Pierre, avec qui j'ai eu le plaisir d'échanger lors de la soutenance ou en conférence ; c'est un honneur d'avoir finalisé ce travail avec vous.

Dans la formidable équipe STORM je compte plus d'amis que de collègues ; Anahid, Florian, nos rendez-vous gourmands sont maintenant digérés mais ils restent au fond du cœur. Chems, Hugo, Nicolas, Olivier et Pierre : ceux qui connaissent la vie de thésard et qui ont su la rendre plus agréable tous les jours. Robin, que j'ai eu la chance de côtoyer quelques (trop courts) mois, et bien sûr Gauthier ; pour son travail acharné sur le plugin Arnold certes, mais surtout pour sa bonne humeur et son amicalité.

Je remercie l'équipe de Michael Wimmer qui m'a reçu à Vienne : Ildar et ses thés savoureux, Markus, Adam, Christian et Hiro pour nos échanges. Merci aussi à l'équipe de la terrasse de l'IRIT, auprès de laquelle j'ai pu me ressourcer à toute heure de la journée.

Agnès, tu as sauvé cette thèse bien avant son début et rien n'aurait été possible sans ton aide indéfectible et ton accueil chaleureux par la suite.

Je remercie bien sûr ma famille, et ceux qui s'y sont substitué à Toulouse : Brigitte, Marie-Jo et Alain. Ma chère Amélie, pour avoir éclairé et embelli ces années. Mes amis de Montpellier, et ceux de Toulouse : Estèle et Thibault, Alice et Benoît, Élise et Dopey. L'ensemble de la Coloc 250, et en particulier Sara pour son aide précieuse le jour J.

À René [Des89]

Résumé

La réalisation des films d'animation 3D s'appuie de nos jours sur les techniques de rendu physiquement réaliste, qui simulent la propagation de la lumière dans chaque scène. Dans ce contexte, les graphistes 3D doivent jouer avec les effets de lumière pour accompagner la mise en scène, dérouler la narration du film, et transmettre son contenu émotionnel aux spectateurs. Cependant, les équations qui modélisent le comportement de la lumière laissent peu de place à l'expression artistique. De plus, l'édition de l'éclairage par essai-erreur est ralentie par les longs temps de rendu associés aux méthodes physiquement réalistes, ce qui rend fastidieux le travail des graphistes.

Pour pallier ce problème, les studios d'animation ont souvent recours à la composition, où les graphistes retravaillent l'image en associant plusieurs calques issus du processus de rendu. Ces calques peuvent contenir des informations géométriques sur la scène, ou bien isoler un effet lumineux intéressant. L'avantage de la composition est de permettre une interaction en temps réel, basée sur les méthodes classiques d'édition en espace image.

Notre contribution principale est la définition d'un nouveau type de calque pour la composition, le *calque d'ombre*. Un calque d'ombre contient la quantité d'énergie perdue dans la scène à cause du blocage des rayons lumineux par un objet choisi. Comparée aux outils existants, notre approche présente plusieurs avantages pour l'édition. D'abord, sa signification physique est simple à concevoir : lorsque l'on ajoute le calque d'ombre et l'image originale, toute ombre due à l'objet choisi disparaît. En comparaison, un masque d'ombre classique représente la fraction de rayons bloqués en chaque pixel, une information en valeurs de gris qui ne peut servir que d'approximation pour guider la composition. Ensuite, le calque d'ombre est compatible avec l'éclairage global : il enregistre l'énergie perdue depuis les sources secondaires, réfléchies au moins une fois dans la scène, là où les méthodes actuelles ne considèrent que les sources primaires. Enfin, nous démontrons l'existence d'une surestimation de l'éclairage dans trois logiciels de rendu différents lorsque le graphiste désactive les ombres pour un objet ; notre définition corrige ce défaut.

Nous présentons un prototype d'implémentation des calques d'ombres à partir de quelques modifications du Path Tracing, l'algorithme de choix en production. Il exporte l'image originale et un nombre arbitraire de calques d'ombres liés à différents objets en une passe de rendu, requérant un temps supplémentaire de l'ordre de 15% dans des scènes à géométrie complexe et contenant plusieurs milieux participants. Des paramètres optionnels sont aussi proposés au graphiste pour affiner le rendu des calques d'ombres.

Abstract

The production of 3D animated motion picture now relies on physically realistic rendering techniques, that simulate light propagation within each scene. In this context, 3D artists must leverage lighting effects to support staging, deploy the film's narrative, and convey its emotional content to viewers. However, the equations that model the behavior of light leave little room for artistic expression. In addition, editing illumination by trial-and-error is tedious due to the long render times that physically realistic rendering requires.

To remedy these problems, most animation studios resort to compositing, where artists rework a frame by associating multiple layers exported during rendering. These layers can contain geometric information on the scene, or isolate a particular lighting effect. The advantage of compositing is that interactions take place in real time, and are based on conventional image space operations.

Our main contribution is the definition of a new type of layer for compositing, the *shadow layer*. A shadow layer contains the amount of energy lost in the scene due to the occlusion of light rays by a given object. Compared to existing tools, our approach presents several advantages for artistic editing. First, its physical meaning is straightforward: when a shadow layer is added to the original image, any shadow created by the chosen object disappears. In comparison, a traditional shadow matte represents the ratio of occluded rays at a pixel, a grayscale information that can only serve as an approximation to guide compositing operations. Second, shadow layers are compatible with global illumination: they pick up energy lost from secondary light sources that are scattered at least once in the scene, whereas the current methods only consider primary sources. Finally, we prove the existence of an overestimation of illumination in three different renderers when an artist disables the shadow of an object; our definition fixes this shortcoming.

We present a prototype implementation for shadow layers obtained from a few modifications of path tracing, the main rendering algorithm in production. It exports the original image and any number of shadow layers associated with different objects in a single rendering pass, with an additional 15% time in scenes containing complex geometry and multiple participating media. Optional parameters are also proposed to the artist to fine-tune the rendering of shadow layers.

Table of Contents

1 Introduction **1**
1.1 The creative use of lighting 1
1.1.1 Intensity and color 1
1.1.2 The role of shadows 2
1.1.3 Discussion 6
1.2 3D animation pipeline 6
1.2.1 Compositing 7
1.3 Outline and contributions of this work 8
2 State of the Art on Light and Shadow Editing **9**
2.1 Painting and sketching metaphors 10
2.1.1 Painting light source parameters 10
2.1.2 Advanced lighting features 11
2.2 Drag and drop interfaces 12
2.2.1 Warping the scene 13
2.3 Goal based methods 14
2.3.1 Perceptual goals 14
2.3.2 Inverse rendering 15
2.4 Light transport manipulation 16
2.4.1 Under simple lighting conditions 16
2.4.2 Global illumination editing 17
2.5 Image decomposition 18
2.5.1 Image abstraction 18
2.5.2 Lighting decomposition 19
2.6 Summary 20
3 Path Space Clustering **23**
3.1 Light transport 24
3.1.1 Radiometric quantities 24
3.1.2 The rendering equation 25
3.1.3 Surface impact 26
3.2 Path integral formulation 27
3.2.1 Definition 27
3.2.2 Estimation 29

3.3 Light path expressions 30
3.3.1 Open Shading Language 31
3.3.2 Implementation 32
3.3.3 Results and applications 33
3.4 Limitations of LPEs in compositing 35

4 Shadow Layers for Solid Objects 37
4.1 Limitations of existing software 38
4.1.1 Overview of our method 39
4.1.2 Involving objects 41
4.2 Shadow from image subtraction 42
4.2.1 The intuition 42
4.2.2 Examples 42
4.2.3 Discussion 43
4.3 Characterization of shadow in the path space 44
4.3.1 Subtraction of the path integrals 44
4.3.2 The set of encountered paths 45
4.4 Efficient rendering of multiple shadow layers 45
4.4.1 Algorithm outline 45
4.4.2 Enhanced user control 48
4.5 Results and performance 49
4.5.1 Application in compositing 51
4.5.2 Performance considerations 52
4.6 Advanced implementation guidelines 53
4.6.1 Performance parameters 53
4.6.2 Shadow layers with other integrators 57
4.6.3 Shadow path expressions in Arnold 58
4.7 Limitations 60
4.8 Conclusion 61

5 Generalized Shadow Layers 63
5.1 Radiance loss along a ray 64
5.2 Generalized path integral formulation 66
5.2.1 Classic expansion with participating media 66
5.2.2 Generalization to the measure of shadow 67
5.3 Equivalence for a single solid caster 69
5.4 Integration in a path tracing framework 72
5.4.1 Key steps 72
5.4.2 Additional parameters 74
5.5 Results and performance 75
5.6 Conclusion 79

6 Conclusion 81
6.1 Summary 81
6.2 Future work 82

List of Figures

1.1 Illustration of the chiaroscuro technique 1
1.2 Two distinct color schemes used to locate the action 2
1.3 Composition and lighting in black-and-white shots 2
1.4 Examples from the storyboard of Ratatouille 3
1.5 The creative use of shadows in Shrek 3
1.6 Discordance between characters and their shadow 4
1.7 Shadows as elements of story and gameplay in video games 4
1.8 The manifestation of shadow in artworks 5
1.9 Influence of shadow on perception and its limits 5
1.10 Detail of the last steps of the pipeline 7

2.1 Direct and indirect light editing 9
2.2 Shadow editing with envyLight 11
2.3 A session of caustics design 11
2.4 Animation of caustics using vector fields 12
2.5 Interactive cinematic shadow design 13
2.6 Drag and drop editing of shadows 13
2.7 Optimized lighting for material depiction 14
2.8 Inverse rendering using differential information 15
2.9 The emission profile of a BendyLight 16
2.10 Shadow transformation with visibility editing 17
2.11 User-controlled appearance of volumetric beams 17
2.12 Duplication of a caustic using RayPortals 18
2.13 Path-space manipulation of a mirror reflection 18
2.14 Image abstraction with Photo2ClipArt 19
2.15 Image relighting with a geometry-aware network 19
2.16 Various filters for stylized shadows 20

3.1 A compositing graph in the Natron software 23
3.2 Rendering of the Cornell Box scene 24
3.3 Geometric setup of the rendering equation 26
3.4 Analysis of impact in three different setups 27
3.5 Caustics and shadow projected by a glass slab 27
3.6 Different event scattering types in the Coffee Maker scene 31

3.7 A four-states language automaton matching caustics 33
3.8 Separation of light sources using LPEs 34
3.9 Advanced compositing using object identifiers 34
3.10 Example of compositing and AOV exports 36

4.1 Result of conventional renderers on the Simple Sphere scene 38
4.2 Indirect shadow missed by the shadow matte of Arnold 40
4.3 Naming convention for objects 41
4.4 Difference between self-shadowing and cast shadows 41
4.5 Shadow layer by subtraction in the Cornell Box 43
4.6 Shadow layer by subtraction on the Fertility Statue 44
4.7 Three examples showing the construction of paths 47
4.8 Shadow layer of the Dragon with and without self-shadowing 48
4.9 Purely specular surface and shadow catching 49
4.10 Common compositing tasks on the Marbles scene 50
4.11 Advanced use of the shadow ratio in the Flowers scene 51
4.12 Shadow editing in the Dragon scene 52
4.13 Lighting balanced in the Moana Island scene 54
4.14 Effect of the maximum number of skips 57
4.15 Our Arnold for Maya implementation 59
4.16 Inconsistencies appearing with multiple occlusions 60

5.1 Shadow coupling with participating media 63
5.2 Different types of volumetric events 64
5.3 Geometric setup of the volume rendering equation 65
5.4 Rendering of the Cornell Box with a participating medium 67
5.5 Indirect shadow from a medium in the Dragon scene 69
5.6 The geometric factor relating area and projected angle 71
5.7 Path construction with two volumetric shadow casters 73
5.8 Shadow removal with and without self-shadowing 73
5.9 Evolution of rendering time with the number of casters 76
5.10 Advanced shadow compositing on the Beach scene 77
5.11 Coherence of shadow ratios in the animated Manhole scene 78

Chapter 1

Introduction

1.1 • The creative use of lighting

While lighting is ultimately the result of a physical phenomenon, it has been explored and manipulated in works of art for centuries. The applied context of this thesis is computer-generated imagery, a domain that experiences this duality. On one hand, the introduction of physically-based rendering has unified movie production pipelines, and allows realistic simulations of light transport. On the other hand, content creators must retain a certain degree of freedom when authoring lighting, as we explain in the following.

1.1.1 • Intensity and color

Lighting design is a multifaceted tool to forge impressions at a glance. Painting leverages light in many forms; for instance, the well-known *chiaroscuro* effect draws the viewer's eyes toward a specific zone of the frame using strong contrasts. The technique was developed during the Renaissance by painters such as Rembrandt and Caravaggio, and is still widely used in motion picture as shown in Figure 1.1. The reason is that, according to Sharon Calahan, *"The primary objective of good lighting is to show the viewer where to look."* [Kah96].



Figure 1.1: *The chiaroscuro technique exaggerates contrast to draw attention toward a precise zone of the scene. Left: The parable of the Rich Fool by the Dutch master Rembrandt (1627). Right: a similar atmosphere from The Incredibles [Bir04].*



Figure 1.2: *Easily distinguishable color schemes from The Lego Batman Movie [McK17].*

In addition to guiding attention, lighting also acts as a conduit for information and emotions. This purpose was formalized in Antiquity by color theory and is a pillar of visual arts [Gur10]. For instance, when cross-cutting is used to alternate between multiple locations during a movie sequence, colors often hint where the action is taking place. In Figure 1.2, we see how the color schemes of the villain and hero are emphasized to quickly associate the setting with each opposing side [Bre20]. Beyond giving practical indications, color also helps the unfolding of a narrative by conveying feelings to the viewer. A classic example of color-emotion associations in animated motion picture is the movie *Inside Out* [Doc15], where they serve as the basis for the story.

1.1.2 • The role of shadows

A natural byproduct of illumination is shadow, that appears when an area is lacking light compared to its surroundings. Shadows are a powerful tool for artistic expression as well, and are even self-sufficient when it comes to storytelling. As an example, shadow puppetry is an ancient form of theatre where the action is played by cast shadows from cut out shapes on a translucent screen. Originated from ancient Asian folklore, the practice of shadow puppetry has been refreshed by contemporary movie directors such as Lotte Reinigier [Rei26] and Michel Ocelot [Oce00, Oce11] using silhouette animation.

Even when they do not embody the entirety of the action, shadows play a full part in the composition of movie shots. Lighting must be carefully set up so that they do not overwhelm the rest of the scene, and receive attention only when intended. An unprecedented level of control was achieved in *The Cabinet of Dr. Caligari* [RW26], where light streaks and shadows were painted directly onto surfaces of the set (Figure 1.3 left).

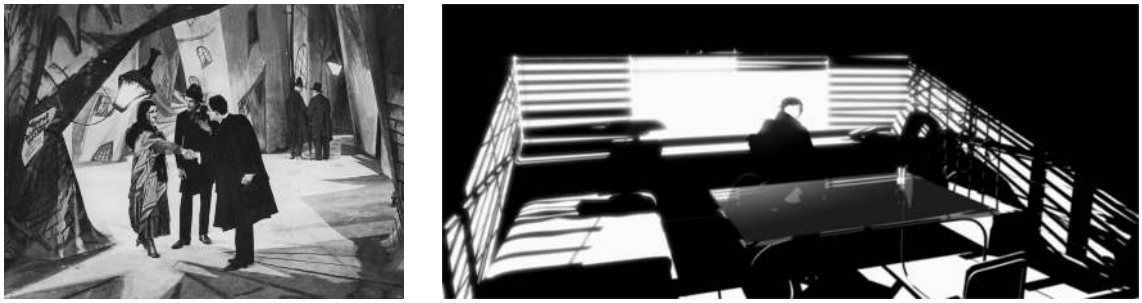


Figure 1.3: *Sharp, angular compositions of light and shadows in these black-and-white shots. Left: The Cabinet of Dr. Caligari [RW26]. Right: Renaissance [Vol06].*

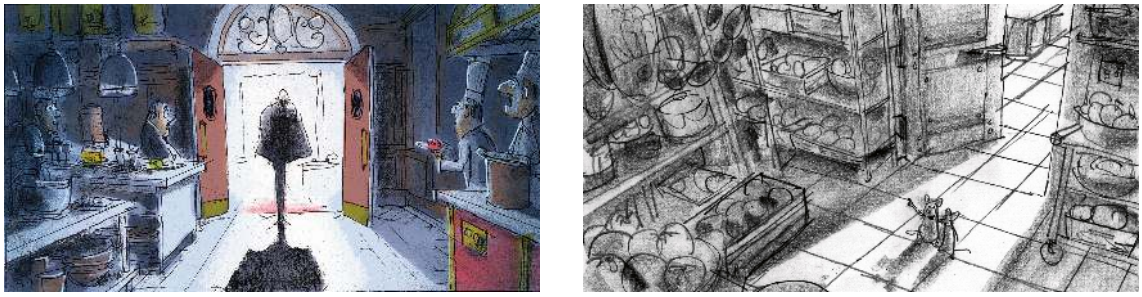


Figure 1.4: *A classic setup: characters step into a dark room through a bright opening [Bir07].*

The *film noir* genre that followed a decade later took inspiration from the experiments of German expressionism, and confirmed the importance of shadows in low-key, black-and-white shots. Since then, a few animated feature films have reconnected with these dark aesthetics; *Renaissance* [Vol06] is a notable example (Figure 1.3 right).

A precise positioning of shadow is all the more desirable as they give many clues on concrete aspects of the scene such as its depth or the motion of objects, but also symbolic ones like the intent of characters or their relationship. In Figure 1.4, we interpret the storyboard of *Ratatouille* [Bir07]. On the left, the fearsome food critic Anton Ego is preceded by a dark shadow that represents his reputation; his irruption in the kitchen reveals the cooks lurking in obscurity. On the right, the rats are about to pilfer food from the storeroom; just like the pointing finger, their shadow announces the plan.

The movie *Shrek* [Ada01] features a particularly creative use of shadows (Figure 1.5). In the torture scene of the Gingerbread Man (left), the repeated soaking into milk is partly seen from the shadow cast next to Lord Farquaad. This efficiently maintains action continuity between shots, that alternate between Farquaad and the executioner. On the right, Donkey dominates the arena from the barrel; this relationship is enforced by the strong overshadowing. Such a lighting is deliberate, and has required some work to break the rules of physics. First, the shadow of Donkey's head is duplicated on both the barrel and the ground. Second, whereas the scene occurs under sunlight, shadows are not parallel and seem to converge toward the top left for the eye to follow the movement. The artists have likely placed a light source for each object, a technique known as *light linking*.



Figure 1.5: *The creative use of shadows in Shrek [Ada01], where they indirectly depict the action (left) or guide the viewer's eyes toward it by breaking physics (right).*



Figure 1.6: *Interesting effects created by the discordance between characters and their shadow. Left: Mulan [Coo98]. Right: The Man With No Shadow [Sch04].*

Sometimes, the difference between characters and their shadow serves the narrative. In *Mulan* [Coo98], the little dragon Mushu uses light from a fire to inflate his silhouette into a terrifying figure, before his real form is revealed (Figure 1.6 left). On the contrary, an old tale from Adelbert von Chamisso narrates the story of a man with no shadow [vC14]: in exchange for a bottomless wallet, Peter Schlemihl has sold his shadow to the Devil, only to find out that this particularity makes him undesirable to others. This story was later animated by Georges Schwizgebel [Sch04] using paint-on-glass (Figure 1.6 right).

The discordance between shadow and the character casting it has also been explored in video games. In *Little Nightmares* [Tar17], the player encounters small gnomes with pointy hats, whose origin is unknown at first. However, the light from a furnace reveals their true form in a later expansion of the game, as seen on the left of Figure 1.7. Judging by the shape of their shadow, it appears that the gnomes were once children. The interactive experience offered by video games is also an occasion to leverage shadows as a core element of gameplay. *In My Shadow* [Pla21] proposes to use them for puzzle solving: from the set of objects found in the level, the player must form a shadow that allows the protagonist to reach the right part of the room (Figure 1.7 right).

The creation of figurative shadow shapes from seemingly random arrangements of objects is a vast topic that has been explored in art, and scientifically studied by Mitra *et al.* [Mit09]. The authors provide a number of examples, and we display two in Figure 1.8.



Figure 1.7: *Two examples of shadow being used as an element of the story and gameplay in video games. Left: Little Nightmares [Tar17]. Right: In My Shadow [Pla21].*



Figure 1.8: *From left to right: Wild Mood Swings (Tim Noble and Sue Webster, 2009); Lunch With a Helmet On (Shigeo Fukuda, 1987); Birdy on shore leaf (Vincent Bal, 2016).*

On the left, the artistic installations of Tim Noble and Sue Webster involve disparate piles of objects; only the light of a projector can reveal their deliberate setup, and cast shadows that depict subjects. In the middle, we see that this technique was also employed by artist Shigeo Fukuda, a master of optical illusions who welded various utensils such as forks and knives to create this piece. In the same spirit, combining hand drawing and cast shadows is a favorite of visual artist Vincent Bal, author of an ongoing series of creations where the two materials complement each other to create little scenes, as shown on the right.

Shadows and perception

A reason that explains the use of shadows in works of art is their implication in the human perception: they are a strong clue in the interpretation of depth and spatial relationships between shapes. On the left of Figure 1.9, two examples from the work of Kersten *et al.* [Ker97] illustrate how the deduced location of an object changes based on the position and the size of the shadow it casts. Many other manifestations of such phenomena have been investigated in the literature [Wan92b]; notably, Casati *et al.* have exemplified the visual implications of shadow based on numerous artworks [Cas00, Cas19].

A common conclusion is that while shadows are deeply involved in spatial reasoning, the human perception is relatively tolerant regarding their physical plausibility. Indeed, shadow is interpreted as a local feature of the object casting it rather than a lighting effect, meaning that inconsistencies between distant shadows go unnoticed [San18]. Moreover, the human eye does not account for the physical correctness of cast shadows, especially at boundaries: softness is only a weak factor of plausibility [Wan92a, Hec14], and shape congruence between an object and its shadow is not necessary either [Sat05].

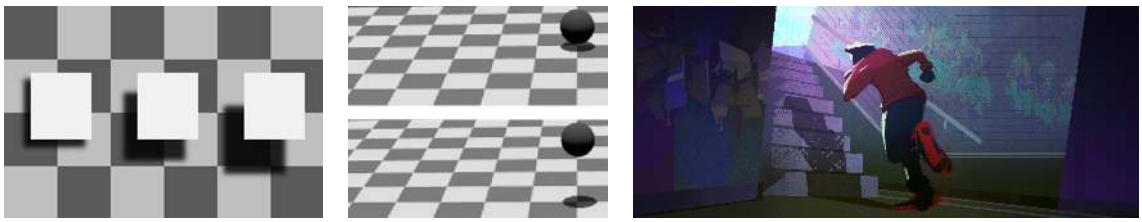


Figure 1.9: *Left: in these two examples, the perception of spatial relationships is strongly influenced by shadow [Ker97]. Right: a physically impossible shadow that does not disturb the eye.*

These conclusions are a motivation of our work, as this perceptual tolerance is known and used in animated motion picture as well. In the color script from *Spider-Man: Into the Spider-Verse* [Lor18] shown on the right of Figure 1.9, we see an effective frame composition where the stance and the shadow of the character inform us on his immediate direction. But judging from the highlights on his clothes and on the wall, most of the illumination is coming from upstairs, and such a shadow is unrealistic. For a lighting artist, reproducing the same shot with simulated illumination is a challenge (in fact, the corresponding frame does not appear in the feature film); we wish to facilitate such tasks.

1.1.3 • Discussion

The previous examples have underlined the role that lighting plays in the creation and depiction of visual arts. We retain three important practices in light and shadow design:

- The intensity and color of light are exaggerated to direct the attention toward a precise zone of the image, or convey information and emotions to help storytelling.
- Shadows play a part in the composition of a frame, and sometimes in the story itself; just as for light, their intensity should be fully controllable.
- Some degree of freedom on the shape of shadows is desirable, and allowed by the human perception. Translation, rescaling, or even repainting create interesting effects.

Our objective is to enable these edits in 3D computer-generated imagery, where the behavior of light is constrained by physics. In order to pinpoint which part of the artistic process we build on, we begin by introducing more context on movie production.

1.2 • 3D animation pipeline

The creation of animated motion picture follows a succession of steps that require very different skills, and thus involve that many teams and departments. Albeit some variations exist, most studios in the industry follow the same artistic pipeline:

Story As for any movie, the creation of animated motion picture starts from a script that contains the dialogues between the characters, and describes their actions.

Art Based on the script, graphic designers elaborate concept arts that capture the appearance of the characters and their environment. They also create a storyboard, a graphical representation of the script that resembles a comic book (Figure 1.4).

Modeling The goal of modelers is to turn the 2D concept arts into faithful 3D shapes. To do so, they usually proceed by sculpting using dedicated software.

Rigging Riggers associate all shapes that undergo deformation with a skeleton, and a set of parameters and controls for convenient manipulation.

Surfaces Surfacing artists define how light interacts with the shape. They must take into account all possible lighting conditions, as surfaces rarely change later on.

Layout The goal of layout is to transpose the intent of the storyboard to compose each set, and bring basic movement to the characters and camera.

Animation Based on the rough trajectories prescribed in layout, animators give believable and evocative movements to all moving shapes.

Simulation Many effects result from simulations: the behavior of clothes, hair, and fur on characters, but also large-scale entities such as crowds and explosions.

Lighting Artists set the position, color, and intensity of light sources in the scene to closely match the *color script* or *color keys* specified during the Art step.

Rendering Rendering is more of a technical step that involves the simulation of global illumination in the scene. This means that multiple bounces between the light sources and the camera are accounted for during the computation.

Post-processing The output of rendering is not the final frame that will be presented in theaters, as many visual effects are achieved in post-processing.

Audio While the pipeline mostly revolves around visuals, the last step is to add music, sounds effects, and voices; the latter are usually recorded beforehand to match lip sync.

We are specifically interested in three stages of the pipeline: lighting, rendering, and post-processing. These steps share some interconnections, as shown in Figure 1.10. Artists alternate between the lighting and rendering steps: they preview the result of illumination, and make corresponding adjustments in a trial-and-error fashion. This process is tedious, as rendering physically realistic light transport involves lengthy computations. For that reason, artists usually save up on time by previewing intermediate results at low quality. When the outcome is satisfactory, a high quality render is exported.

1.2.1 • Compositing

The output of rendering is not limited to a single image. Alongside the result of global illumination (the *beauty* render), lighting artists export other layers called *Arbitrary Output Variables* (AOVs). These layers contain information extracted from the scene, such as the color of surfaces or the curvature of the geometry. In a sense, they incorporate information from the previous steps of the pipeline in image space. This also applies to the rendering step, and AOVs typically contain information on illumination that was extracted from the simulation of light transport. For example, a lighting artist can isolate the illumination on the face of the main character inside a separate AOV.

The beauty render and all AOVs serve as inputs to the compositing stage, where they are assembled to obtain the final frame. The main appeal of compositing compared to the previous steps is that the interaction between the artists and their material takes place in 2D, and real time. Coming back to our example, a simple change in intensity increases the illumination of the main character's face, creating a chiaroscuro effect (Figure 1.1).

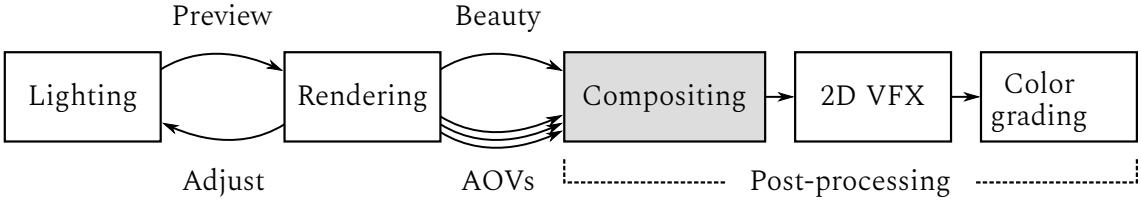


Figure 1.10: *The last steps of the visual pipeline are lighting, rendering, and post-processing. Artists alternate between the first two in a trial-and-error fashion, which can be tedious.*

Post-processing also includes 2D VFX where various effects such as lens flare or particles are added, and color grading that enhances the palette in each frame.

Driven by its convenience and expressive power, compositing is the step of the pipeline on which our work relies to improve the artistic editing of light and shadows. As detailed below, our contribution is a new type of AOV specifically designed to improve the artistic editing of shadows in compositing.

1.3 • Outline and contributions of this work

Having established the applied context of our work, we begin by reviewing the related scientific literature Chapter 2. Among the existing methods that propose to edit light and shadows in computer-generated imagery, few apply to global illumination, and only *Light Path Expressions* (LPEs) allow real-time editing of lighting features.

To understand the inner workings of LPEs, Chapter 3 lays the mathematical and computational foundations of global illumination rendering with Monte Carlo integration. The latter is based on a random exploration of the path space, the space of all light paths that can be formed in a scene, and we reformulate LPEs as describing clusters of light paths that belong to similar lighting patterns. By separating the illumination of these clusters from the rest, we obtain a set of layers that enable advanced compositing of global illumination effects. We demonstrate their use on several scenes, based on our own implementation of the method described in a related communication [Des18].

One limitation of light path expressions is that they cannot account for the formation of shadows, that are a byproduct of the simulation and never explicitly carried by paths. To remedy this shortcoming, our first contribution is the definition of *shadow layers* in Chapter 4. Shadow layers measure the amount light lost on surfaces of the scene due to the presence of a single solid occluder during the propagation of paths. When the shadow layer of an object is added to the original image, any shadow it casts is removed from the scene, and shadow can thus be transformed separately before being re-injected in the composite result. We propose an efficient rendering algorithm that exports any number of shadow layers and the original image in a single pass, with an affordable overhead. Its implementation is carried out in both pbrt-v3 [Pha16] and Arnold for Maya. This chapter corresponds to a Eurographics Symposium on Rendering publication [Des19].

This first approach has two shortcomings however: first, occluders are considered in isolation from one another, meaning that light lost due to different objects along a path is not picked up in any shadow layer. Second, the method on applies to solid objects, leaving out participating media such as clouds and liquids. In Chapter 5, we overcome these two limitations and generalize shadow layers to arbitrary sets of either solid or volumetric occluders. Our formulation does not make assumptions on the properties of the objects, and thus applies to production-grade setups. We demonstrate this claim on several scenes, where our prototype exports all shadow layers and the original image with a controllable overhead and increased efficiency when exploring the path space. The content of this chapter was published in the Computer Graphics Forum journal [Des21].

Chapter 2

State of the Art on Light and Shadow Editing

The reason why numerous edits are necessary to achieve the desired outcome is that light and shadow effects are indirect products of a complex physical simulation, and 3D artists have to alternate between the lighting and rendering steps of the movie production pipeline in a trial-and-error fashion.

- Most of the lighting, surface, and volumetric models are based on physical equations and properties that have little to no artistic value.
- The result of global illumination, and thus the look of a scene, is hardly predictable as it involves multiple bounces between the light sources and the camera.
- Obtaining a faithful render takes some time, and slows down the process of judging the result of a modification before bringing a new one.

Despite these impediments, lighting artists are given a precise artistic goal for each shot based on the color script elaborated in pre-production. This chapter describes a number of approaches that propose to overcome the challenges of lighting design. We distinguish two classes of methods, displayed in Figure 2.1: indirect and direct methods [Sch16].

In indirect methods, the user expresses the desired lighting changes in image space, starting from an initial result. The performed modifications are then propagated back to the scene, typically by using some form of optimization to change its parameters such as the color or the position of a light source. A pointing device can be used for most of the operations, either by painting lighting features (Section 2.1), or dragging and dropping primitives (Section 2.2). Using goal based methods (Section 2.3), lighting is optimized according to perceptual metrics, or by inverse rendering from an objective image.

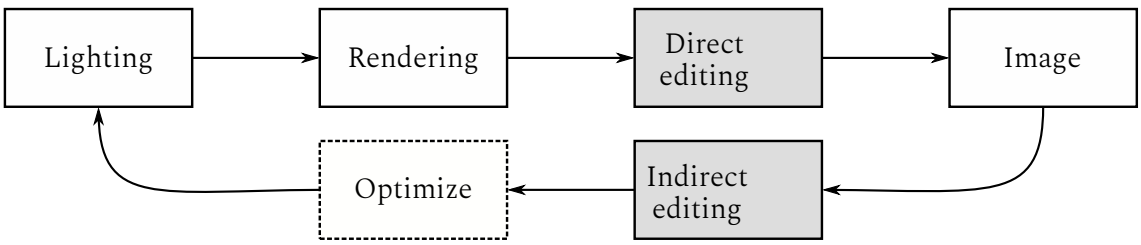


Figure 2.1: *The technical differences between direct and indirect light editing methods are due to their relative position with respect to the rendering step.*

Direct editing methods are different in that they must have some relationship with the simulation algorithm to operate. Some methods are tightly coupled with the rendering process itself and allow the user to manually influence light transport (Section 2.4), while other approaches use insights on the formation of lighting features to offer a decomposition of images into meaningful layers that simplify editing (Section 2.5).

2.1 • Painting and sketching metaphors

Among the users that face lighting design tasks, most are familiar with drawing, painting, or using a pen tablet. For that reason, many approaches rely on a pointing device to paint strokes or sketches as an input [Ker09]. The interaction takes place over an initial image, and the requested changes are applied to the scene by an optimization process hidden from the user. After the desired modifications have propagated to the scene parameters, the user can re-iterate their edits and further tune the final aspect of the rendering.

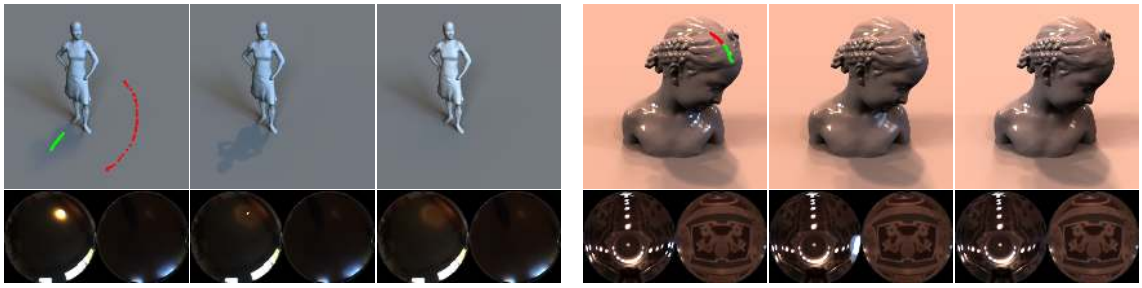
2.1.1 • Painting light source parameters

Brush strokes can be translated into a set of linear equations that determine the color of light sources [Sch93]. Thanks to an efficient implementation, least squares solving of the system and render updates are achieved at interactive speed. However, this technique only works with *radiosity* [Gor84], a rendering algorithm that is limited to depicting perfectly smooth surfaces.

Specified by sketches, the location of highlights and shadows is a strong clue to determine the position of light sources [Pou97]. The user indicates the presence of a shadow or highlight at 3D points and can also add *anti-sketches* to prevent their existence, all in real-time. Again, these constraints translate into a set of equations; the main difficulty here is their non-linearity, which requires an involved optimization procedure [Law99]. Another shortcoming of the method is its limitation to direct illumination, meaning that light interacts only once with objects of scene.

In Lighting with Paint [Pel07], brush strokes are complemented by an importance map to select the most significant features in the image. The non-linear optimization process is then constrained to prioritize those features when computing the lighting conditions. The method adds new lights after each input to reach the desired goal, leading to an uncontrollable and potentially unbounded number of sources. As discussed by the authors, it is possible to remove some of the light sources and optimize the rest to obtain a similar illumination, at the price of additional user intervention.

A common assumption is to consider that incoming light is dependent only on the direction. In this case, indirect methods automatically determine an environment map that encodes the desired incident light distribution. Using the Illumination Brush interface [Oka07], the user is able to create an environment map using two primitives: a diffuse brush for a smooth and uniform aspect and a specular brush to create sharp and view-dependent reflections. The algorithm uses spherical harmonics to populate the map interactively and efficiently render the scene [Slo02].



Selection, sharpening, and removal of a shadow. *Selection, blurring, and removal of a highlight.*

Figure 2.2: *The envyLight interface [Pel10] allows the selection of lighting features using sketches. Top row: selection sketches and rendered result. Bottom row: the transformed environment map.*

The envyLight interface [Pel10] is designed to edit preexisting environment maps, and sketching serves as the basis for the selection of lighting features (Figure 2.2). After a selection, the algorithm separates the environment map into a foreground layer to perform the modifications, and a background layer that is left untouched. Various tools are provided to transform the foreground layer such as translation, blurring, or sharpening.

2.1.2 • Advanced lighting features

The appearance of highlights is very difficult to author, as they result from view-dependent interactions between light and the surface material. By setting up a local drawing frame over an object, the user can directly sketch the appearance of highlights before projecting them onto the surface [Pac08]. The interaction takes place at fixed lighting and viewing directions, but the shape of highlights is represented as a polar curve that is interpolated under unknown light and camera configurations. The projection of highlights assumes objects of low curvature with smooth surface normals; if these conditions are not met, the coherence of the highlight deformation is not guaranteed.

While the method of Nguyen *et al.* [Ngu13] primarily transforms surfaces, it lets the user design the expected lighting by performing strokes over objects of the scene. This input is interpreted as a set of constraints for the *surface light field*, which represents the amount of light outgoing at every direction of a surface point. The interface supports two editing modes: in direct mode, the considered surface light field is located directly under the pointer, while the indirect mode instructs the algorithm to target a surface light field at another location. The example editing session in Figure 2.3 shows the creation of a caustic, an advanced lighting feature that is easily added here using the indirect mode.



Figure 2.3: *Creation of a caustic from two input strokes [Ngu13]. To chosen interaction mode is indirect, which instructs the algorithm to optimize lighting away from the pointer location.*

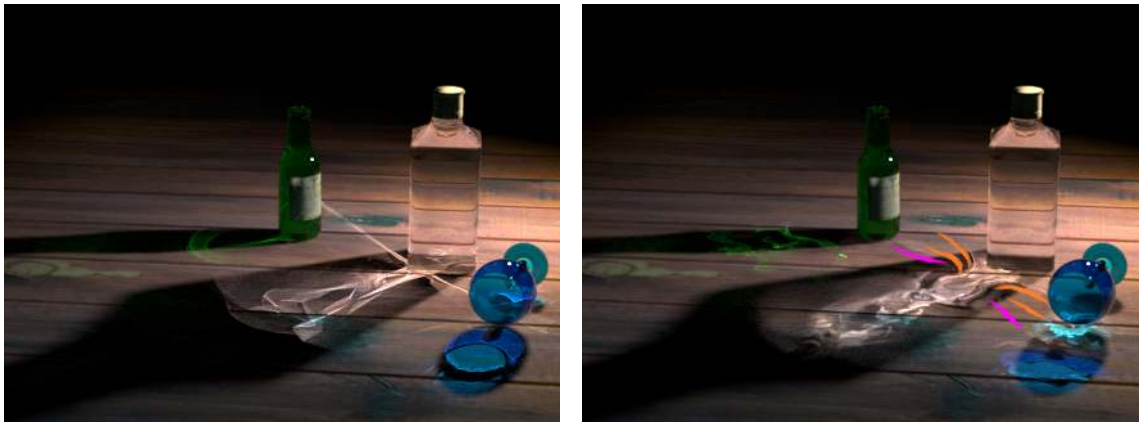


Figure 2.4: *The original caustic (left) is animated using a set of strokes (right) [BR20]. Walls drawn in magenta prevent photons from crossing, while orange directed lines guide their motion.*

Caustics are intricate patterns of light created by the reflection or refraction of light on curved, highly specular surfaces. Their stylization has been addressed based on a drawing metaphor in the work of Baeza *et al.* [BR20]. Thanks to the rendering framework of photon mapping [Hac09], their approach keeps track of the photons forming the caustic and treats them as particles moving in a fluid. To control the behavior of the fluid, the user draws either directed lines to guide its movement, or wall lines that prevent particles from crossing (Figure 2.4). The fluid behavior applied to the photons requires a lengthy simulation, leading to computation times of 3 to 11 hours for 100 animated frames.

Other advanced illumination effects such as participating media can be stylized using a painting metaphor [Kle14]. The artist guides the underlying optimization with a set of images that depict the desired medium appearance under different points of view. Those images are obtained by rendering an initial, unstylized medium, and transforming the different views using a third-party image editing software. Based on the pixels of the reference images, the algorithm infers the properties of the medium at each voxel.

2.2 • Drag and drop interfaces

Another form of interaction that relies on pointing devices is the drag and drop metaphor. It enables intuitive translation and rescaling of shadows and light hotspots across surfaces of the scene, as seen in Figure 2.5 [Pel02]. The system also supports the addition of constraints to assist in the transformation of the selected features.

In a related work [Pou92], the user positions the center of circular highlights, and moves the cursor until the desired radius is met. This information is sufficient to determine the direction of the sun light primitive that creates the specified highlight [Han90], but also the roughness of the pointed surface. The second contribution of this work is the automatic creation of light sources from shadows. Contrarily to highlights, the location and appearance of shadows does not depend on the point of view, leading to more predictable results. While two interactions are enough to instantiate a point light from the shadow it casts, more inputs are required for advanced emitters such as polygonal lights.

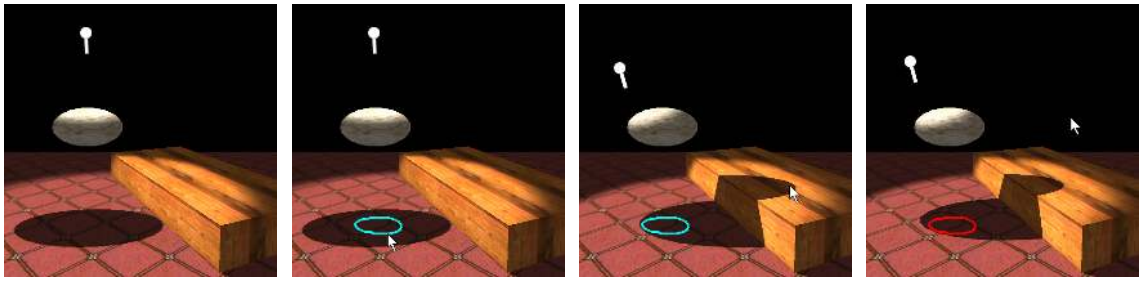


Figure 2.5: *An example of indirect editing using drag and drop [Pel02]. The user moves an object's shadow using the cursor, and the edit is constrained by the colored circle placed beforehand.*

In Freeform Shadow Boundary Editing [Mat13], the primitives being dragged are not lighting features, but control points laid out around cast shadows (Figure 2.6 left). Every time they are tweaked, the system updates a *shadow mesh* whose purpose is to cast the intended shadow. It is also possible to use functions to displace the position of the control points, using for example a sinusoid to produce a wavy appearance.

The previous approaches only apply to scenes where the relationship between the light sources and the lighting features is simple enough to be inverted. In particular, this implies that rendering is limited to highlights and shadows created by direct illumination.

2.2.1 • Warping the scene

A local warp of the 3D space in which the scene is rendered helps author its final appearance. It allows global illumination effects that appear with multiple interactions between light and objects to be edited using the drag and drop metaphor.

Normally, the appearance of specular materials is hardly customizable because of the constraints they impose on light transport: energy is scattered in a single, deterministic direction. Still, these strict rules can be bypassed by creating and manipulating control points over a chosen surface [Rit09]. The user adds their own set of constraints on light behavior, and the algorithm tries to satisfy them using a least squares minimization scheme. The idea of locally warping the scene space was later extended to drag shadows, caustics, and textures over surfaces [Rit10], as shown right of Figure 2.6.

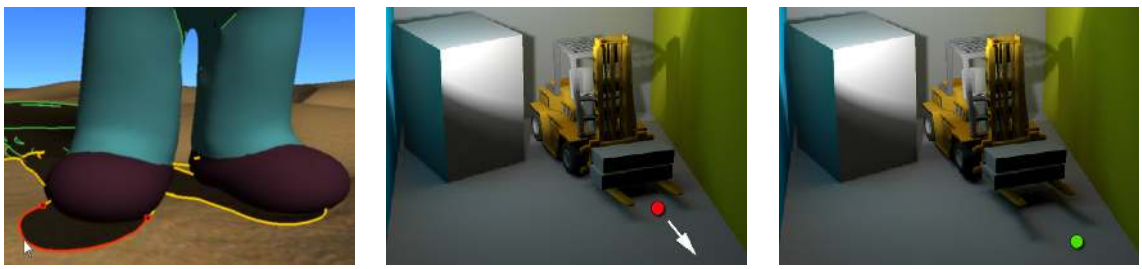


Figure 2.6: *Left: control points on the boundary of shadows help define their shape [Mat13]. Right: shadows can be dragged in the scene thanks to a local warping of space [Rit10].*

The technique of Anjyo *et al.* [Anj06] is not concerned with photorealistic effects, but focuses on highlights design for cartoon animation. Several operations such as translation, split, or shape squaring are expressed as vector field transforms applied over the surface of objects. The user chooses one of them, and the drag and drop interaction begins.

2.3 • Goal based methods

Rather than trying to provide toolboxes and interactive interfaces for the artistic manipulation of lighting, goal based methods aim to describe and modulate the lighting conditions using perceptual quantities and metrics, or simply an input image that serves as a target to perform inverse rendering.

2.3.1 • Perceptual goals

Starting from the results of perceptual studies, Kawai *et al.* have decomposed lighting into specific impressions such as *clear*, *pleasant*, or *private* [Kaw93]. Their system allows the user to assign weights to each of these quantities depending on their intention; the result becomes the objective function of the optimization process.

Given a metric that properly describes what a good lighting consists in, the design space can be explored fully automatically. Image analysis quantities (gradients, color histogram) have been combined with scene information (edges, shading) to obtain a thorough perceptual quality metric [Sha01]. The algorithm balances the various terms of the metric to produce a visually pleasing illumination without any intervention of the user.

The metrics defined by Bousseau *et al.* [Bou11] are directly related to the type of material that is being depicted: for instance, fabric and transparent materials need to receive different illuminations to best reveal their underlying geometry and color. By combining the different metrics, the algorithm automatically explores the design space and proposes an optimal depiction of materials (Figure 2.7). The method is limited to environment maps, as the procedure uses spherical harmonics to optimize emission profiles.



Figure 2.7: *The render on the left lacks details; the optimized lighting in the right image emphasizes the translucency of the candle, and the highlights of porcelain and chrome surfaces [Bou11].*

2.3.2 • Inverse rendering

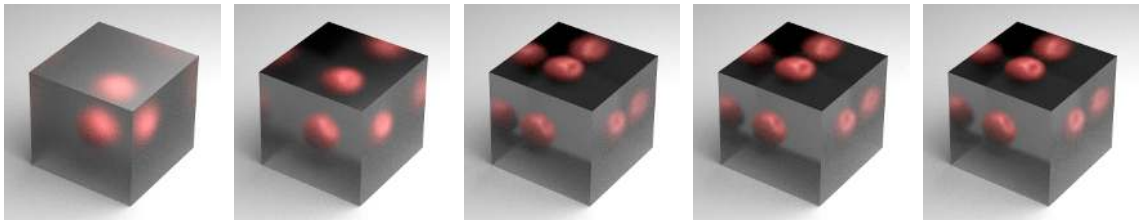
The principle of inverse rendering is to optimize the scene according to a precise illumination goal, given at locations in the scene or directly by the pixels of a target image.

Semi-automatic lighting design was achieved by specifying the amount of light that should be arriving at positions in the scene using *inverse luminaires* [Cos99]. The method is based on the invertibility of light transport: the quantity being propagated is importance, which flows from the camera to the light sources. As the result of global illumination is dependent on complex factors, proper realization of the given goal is not guaranteed. Furthermore, the complexity of the task at hand implies that the optimization process takes several hours to determine the adequate lighting setup.

The same idea was specialized to highly occluded environments [Gka15]: starting from a discrete uniform initialization of light sources, a mutation strategy is applied to progressively reach the desired illumination while taking into account intricate visibility effects. This approach is tailored to lighting design for buildings interiors, where the layout of rooms is the main factor of occlusion.

Several methods have been presented to design real optical systems that project a given image as a caustic when illuminated [Pap11, Sch14], or to insert procedural caustics into real images [Gut08]. The idea of projecting custom patterns as caustics has been transposed to synthetic images as well [Gün16]. Once again, photon mapping is used to keep track of the photons that describe a caustic. The user provides a target image, and the algorithm is able to interpolate the caustic between its original form and the target image. This is achieved by organizing photons in small groups, and smoothly transitioning their emissive properties. However, the method performs poorly when the surface receiving the caustic is curved, and does not extend to animated illumination conditions.

Directly related to goal based methods, differentiable rendering algorithms are able to generate the gradients of image pixels with respect to scene parameters. This information is used to perform inverse rendering by optimizing the scene according to a target image (Figure 2.8). Early considerations of differential quantities in rendering consisted in determining pixel footprints for antialiasing [Ige99]. Later work by Ramamoorthi *et al.* has shown that shadows can be defined as the derivative with respect to visibility [Ram07].



Initial state *Iteration 10* *Iteration 40* *Iteration 80* *Target image*

Figure 2.8: *Differentiable rendering is an important tool for inverse rendering, the optimization of scene parameters according to a target image [Zha19].*

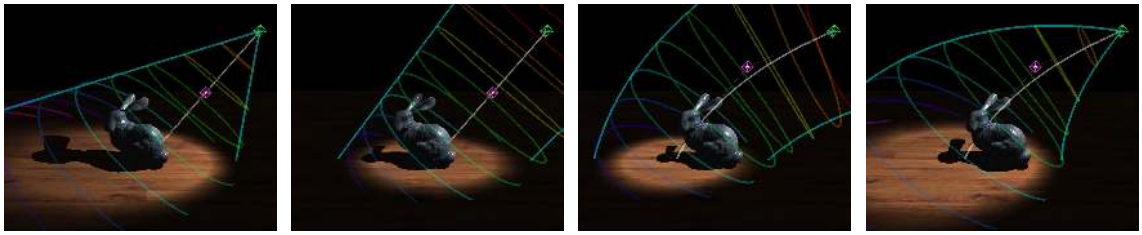


Figure 2.9: *A BendyLight [Ker10] is a variant of the spotlight primitive that offers precise control of its emission profile using a set of Bézier splines.*

When transposed to Monte Carlo simulation, differentiable rendering is particularly challenged by well-known difficult light transport phenomena such as the complicated patterns formed by caustics and fine geometry, but also encounter new issues. Most notably, differentiating the light transport equations introduces integrands with strong discontinuities where visibility varies. To circumvent this, Zhang *et al.* have chosen to operate directly on integrals involving full light paths instead of differentiating the propagation equations [Zha20, Zha21] Dedicated schemes were also designed to sample area [Lou19, Ban20] and boundary edges [Li18, Zha19]. The main limitation of differentiable algorithms is their computational complexity; a large body of work is dedicated to reducing the required processing power and memory, while maintaining derivatives over large sets of parameters [ND20, Vic21]. Despite this extensive literature, many questions remain open regarding the mathematics of differentiable rendering [Zel21].

2.4 • Light transport manipulation

While light transport is a complex phenomenon, letting the user rework it directly enables more precise editing of illumination. In the following works, technicalities are abstracted behind usable interfaces that also allow physically impossible results.

2.4.1 • Under simple lighting conditions

BendyLights [Ker10] presents a variant of the spotlight primitive, where the emission profile can be deformed along Bézier splines to retarget or fine-tune direct illumination. As seen in Figure 2.9, the user controls the trajectory that light rays follow, but also the diameter of the spot at each segment. Technically, the method works by deforming the whole rendering space according to the inverse of the spotlight transform. For this reason, its performance hardly scales in the presence of multiple light sources as the geometry must be re-projected for each transform.

Shadows can be transformed in a non-physical way by altering the visibility of objects under direct environment map illumination [Obe10]. A wavelet framework allows efficient encoding of the visibility without discarding its high-frequency features. The authors introduce a meaningful quantity, the *shadow ratio* between the original color of the rendering and the color after shadows have been removed. While visibility itself is a binary information, visibility ratios are RGB components that provide an intuitive understanding of occlusion effects, and serve as the basis for manual editing as shown in Figure 2.10.



Figure 2.10: *By altering the visibility of objects [Obe10], the user transforms the original render (left) to recolor shadow (center), or applies custom patterns (right).*

2.4.2 • Global illumination editing

While global illumination rendering consists in simulating complex physical equations, some works have focused on introducing better artistic control in the process.

Nowrouzezahrai *et al.* [Now11] address the artistic direction of volumetric lighting. By reworking the equations that define a participating medium, they show that the intricate physical properties that describe emissive volumes can be turned into four meaningful and controllable quantities. Thanks to this re-parameterization, the user applies custom profiles to light beams with the guarantee of a predictable outcome (Figure 2.11).

The iCheat framework [Obe08] enables local control of light transport. Rendering is formulated in terms of the transport coefficients between two points: $T(v, g)$ is an RGB vector that describes the amount of light transported between a gather point g and view point v. The user rescales and offset these coefficients as $T'(v, g) = s(v, g)T(v, g)+o(v, g)$, and the edits are stored inside a wavelet-compressed 6D structure.

Instead of reworking light transport locally, RayPortals [Sub17] enable the teleportation of rays across the scene. Portals are manually set up as pairs of surfaces in the scene that share a common parameterization, and their influence can be limited to a subset of paths based on the nature of previous interactions (diffuse or specular). Additionally, the interface of a portal can change the intensity and hue of light rays, or modulate the propagation direction. The duplication and transformation of a caustic is shown in Figure 2.12.

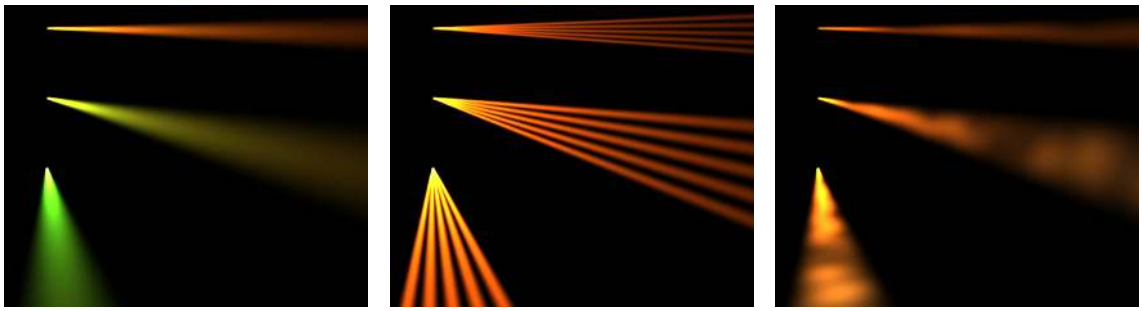


Figure 2.11: *User-controlled appearance of volumetric beams [Now11]. Different profiles are obtained by composing familiar functions that depend on only four meaningful parameters.*



Figure 2.12: *The RayPortals interface [Sub17] is capable of replicating and transforming the appearance of advanced lighting features, such as a caustic here.*

Schmidt *et al.* [Sch13] also propose to work directly with light paths, thanks to a visualization and selection technique. The user picks a group of paths with similar properties, and retargets them via linear transforms (Figure 2.13), or using objects proxies to influence their propagation.

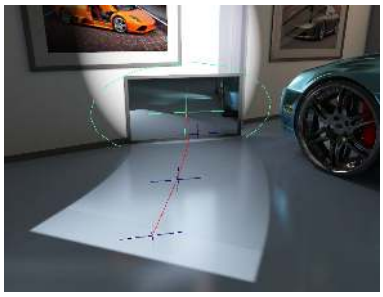


Figure 2.13: *Manipulation of paths on a reflection [Sch13].*

The main liability of the previous methods is the global illumination framework they operate in. While its allows the design of advanced lighting features, an expensive re-rendering of the scene is required to assess the result of every edit.

2.5 • Image decomposition

A simple solution to achieve fast editing of lighting is to perform the operations directly on the rendered images. While the artistic opportunities get more limited using conventional tools, a proper decomposition of the image yields a broad creative space.

2.5.1 • Image abstraction

The decomposition of images into a set of simple primitives provides an abstraction over their content that empowers artistic editing. Diffusion Curves [Orz08] is a framework to represent images as a set of Bézier splines from which emanate color gradients. An input bitmap image can be automatically converted into this representation without losing much of its details and sharpness. This process is useful for further processing and animation, as the image is now generated from a set of easily keyframed parameters.

The human eye is well-trained to interpret the overlapping of semi-transparent objects. This capability has been replicated in software [Sin03] based on the detection of X junctions [Kas99], the points along the edge of a semi-transparent object that intersect a change of color below it. The detection and partitioning of transparency and highlight effects into separate layers was ported to commercial image editing software [Ric14], where it augments the possibilities of the *magic wand* tool. The user selects a semi-transparent or opaque region of the image, and the system figures out a decomposition into a foreground gradient and a background bitmap layer. After a few interactions, the image is represented as a superposition of layers that enable a non-linear workflow.



Figure 2.14: *Photo2ClipArt [Fav17] provides image abstraction at the cost of minimal user intervention, with an output vector representation that is well-suited to perform manual edits.*

While it achieves the same type of decomposition, Photo2ClipArt [Fav17] minimizes manual user intervention by performing an extensive exploration that balances between the fidelity and simplicity of the decomposition. Presented in Figure 2.14, the result of the procedure is an abstract version of the original that greatly simplifies manual edits: a change of color in a bottom layer remains coherent with the rest of the image.

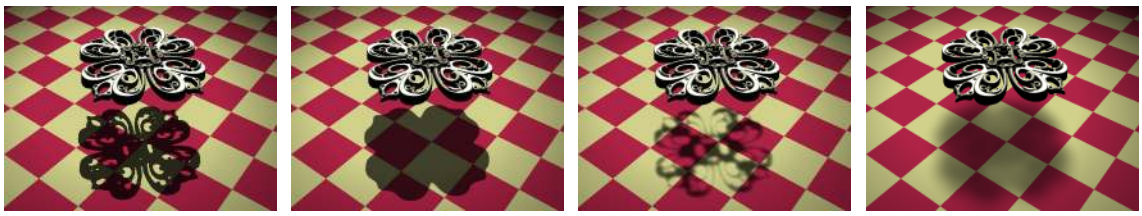
2.5.2 • Lighting decomposition

Intrinsic decomposition is a vast field of research [Bon17] whose general goal is to express a given image as a product of two terms: the diffuse surface albedo and the illumination. More precise decompositions of the lighting term have been studied for outdoors scenes, taking into account the sun, sky, and indirect illuminations [Laf13]. Newer iterations of intrinsic decomposition methods have achieved faithful separation of global illumination in videos [Mek21] and multi-view scenes [Duc15].

Whereas lighting components are additive with each other, their image space compositing requires more involved considerations. Indeed, the geometry of the scene induces warping on the lighting features: when a shadow is cast over a sphere, it should follow its curvature for a convincing result [Wil78, Chu03]. Those considerations are particularly important when the lighting transformation is assisted, such as in relighting tasks. Neural networks have proven to be efficient for this purpose; they allow easy compositing of extraneous objects into real environments displayed with image-based rendering [Nic20]. By coupling a network with information on the geometry of scene, multi-view relighting is achieved while respecting the projection of shadows [Phi19, Phi21] as seen in Figure 2.15.



Figure 2.15: *From the first image on the left, a geometry-aware neural network produces convincing relit images that respect the behavior of cast shadows [Phi19].*



Original shadow *Shape abstraction* *Shrunk and softened* *Blurred and lightened*

Figure 2.16: *By using information from the rendering process, the shape of shadows can be efficiently transformed in image space using predefined filters [DeC07].*

While the previous approaches to lighting decomposition apply to photographs, synthetic imagery can benefit from computational insights about the formation of lighting features in a scene. Under direct illumination, shadowing algorithms allow the extraction of visibility as a binary map that determines if a pixel receives the contribution of each light source. DeCoro *et al.* [DeC07] have leveraged this visibility information as the input of a set of predefined image space filters. As shown in Figure 2.16, these filters open a broad creative space to transform the shape and appearance of shadows in real-time.

The editing of images obtained with global illumination algorithms is also enhanced by information from the underlying simulation process. In movie production, lighting and compositing artist now widely rely on light path expressions [Bre20] to manipulate lighting after a scene has been rendered. Light path expressions are akin to regular expressions, and allow the decomposition of a synthetic image into multiple layers. For instance, lighting due to diffuse and specular surfaces can be split inside distinct layers, and then reworked separately. This technique has the upside of offering real-time will manipulation of global illumination, and will be presented and discussed thoroughly in Chapter 3.

2.6 • Summary

In order to assist the design of light and shadows in synthetic images, we have seen that two strategies had prevailed in previous work.

On one hand, editing metaphors such as painting or drag and drop are reminiscent of physical gestures that graphic designers are familiar with. Yet in the implementation, this simplicity often transfers to the illumination conditions: rendering may be limited to direct lighting, or surfaces and light sources restrained to a given type. Intuitively, the weak constraints that simple interfaces enforce on the design space must be offset by strong assumptions on the scene and the illumination model to avoid ambiguities.

On the other hand, some approaches try to minimize user intervention by automatically optimizing the scene according to fixed goals. The interest and recent advances in differentiable rendering have confirmed the momentum of inverse rendering methods. A strong restriction of these tools is that they omit a fundamental step in the artistic process: the exploration of the design space. In animated motion picture, the goal of a shot is indeed known beforehand, but as a very abstract set of expectations that relate to emotions and storytelling rather than radiometric constraints.

As we aim to create a permissive design tool for global illumination, only a few methods relate to our objective, mostly from Section 2.4.2. However, we have already underlined their main limitation: re-rendering the scene is required after each edit, which prevents real-time interaction. With this additional constraint, image decomposition using light path expressions comes closest to our goal. This explains its widespread use in studios, and why we build on this idea in the following.

In Chapter 3, we begin with a brief introduction to the theory behind realistic rendering and the simulation of light paths inside the scene. This first step is necessary to understand that light path expressions are a tool to describe clusters of light paths based on their propagation history, whose contribution in image space is exported separately to serve in compositing. While they cover intricate lighting features such as caustics, light path expressions are not aware of the presence of shadows. Chapter 4 thus introduces *shadow layers*, a method that shares the same motivation but enables the rendering of shadows inside separate images. In this first formulation, the export of shadow is limited to solid objects, that are considered in isolation from one another. These two restrictions are lifted in Chapter 5: we generalize the method to shadows created by participating media, and cover the case of light paths occluded by multiple objects.

Chapter 3

Path Space Clustering

Post-processing is an important step in the creation of synthetic 3D animations (Section 1.2.1). It takes places after the scene has been rendered, and the resulting image undergoes an extensive set of modifications in order to reach the artistic goal expected from the shot. In particular, the compositing stage brings together multiple layers alongside the original image to create the final frame. These *Arbitrary Output Variables* (AOVs) contain any useful information extracted from the scene. For example, exporting the depth at every pixel of the image inside a separate AOV allows a compositing artist to operate in a 2.5D space by un-projecting pixel locations. This example is typical of compositing: the purpose of this stage is to perform advanced relighting operations, while retaining the comfort of real-time feedback. Compositing edits are described as image space operations and filters usually organized in acyclic graphs, offering a non-linear workflow that naturally handles sequences of frames. An example of such a graph is shown in Figure 3.1.

This chapter presents a set of techniques to provide compositing artists with AOVs that isolate lighting features from one another to empower post-processing. These features are

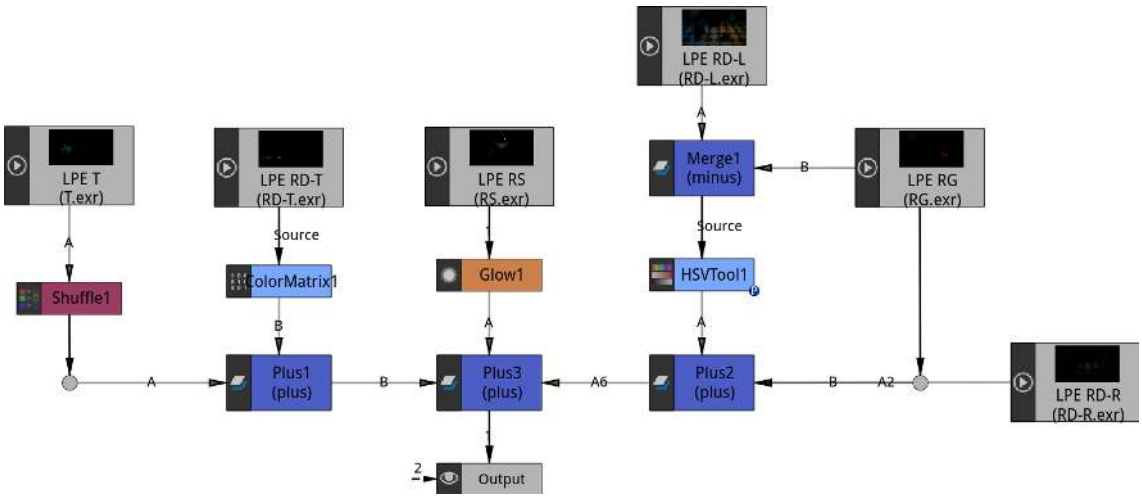


Figure 3.1: *A compositing graph in the Natron 2.4.0 software. The gray nodes in the top and right parts represent the input layers, and the colored nodes are operators and image space filters.*

produced by the realistic rendering of global illumination, and so we briefly introduce the physical background of light transport in Section 3.1. Based on this theory, Section 3.1.3 presents the notion of *impact*, a quantification of how surfaces in the scene influence the distribution of light. Impact is the starting point to generate our AOVs as it describes the apparition of shadows, reflections, and other lighting effects.

However, impact is localized on surfaces and must be translated in image space to be usable; the path integral formulation presented in Section 3.2 is the framework of choice to do so. Its integration domain is the high-dimensional space of all light paths in the scene, the path space. It is possible to create clusters in this space to refine impact, and separate different lighting features: Section 3.3 introduces light path expressions, a formalism to describe clusters based on the propagation that light paths follow. In Section 3.3.2, we present our implementation of light path expressions to extract the contribution of each cluster separately, and output the desired AOVs alongside the original image.

3.1 • Light transport

In order to account for multiple bounces between the light sources and the camera and obtain advanced lighting effects, the realistic simulation of illumination of is based on the physical quantities and equations that rule light transport. The following is a brief presentation of the required theoretical background focused on scenes composed exclusively of surfaces, under the simplifying assumptions of geometrical optics. Advanced effects explained by wave or quantum optics are thus not covered.

3.1.1 • Radiometric quantities

The heart of our problematic is the manipulation of color in synthetic images. In the strict sense, color is the perception of light by the human eye, and originates from the stimulation of photoreceptor cells by light rays. Yet, we use the word color in a broader sense to refer to the distribution of luminous energy in an image, or at a sensor. The term sensor itself encompasses different entities that react to incoming light: the camera film, an electronic capture device, or a light meter to name a few. The goal of rendering is to estimate color at every location of a *virtual* sensor, as illustrated in Figure 3.2.

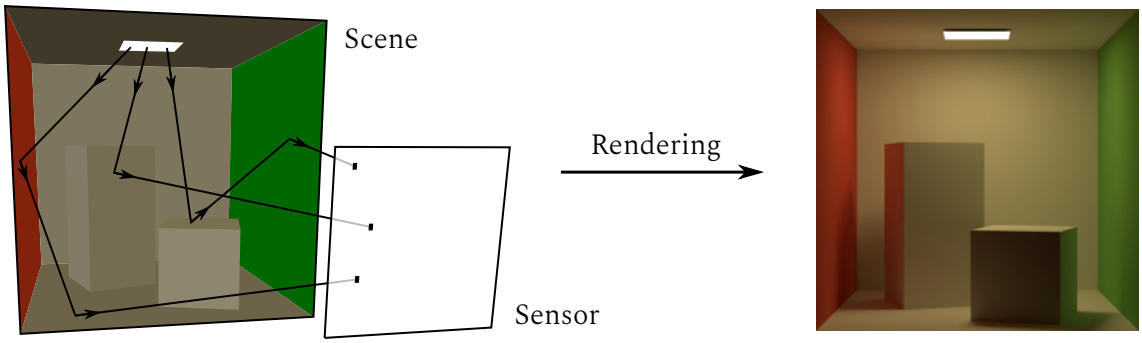


Figure 3.2: *The global illumination rendering algorithms our work is based on estimate color over the sensor by building light paths inside the scene and computing their overall contribution.*

Irradiance When describing color over the sensor, we consider the luminous energy Q deposited by photons on its surface. Each photon transports a quantum of energy proportional to its frequency ν, given by the Planck-Einstein relation: $q_\nu = h\nu$, with h the Planck constant. Irradiance E ($\mathrm{W.m^{-2}}$) is the total energy received from photons during an infinitesimal time interval $\mathrm{d}t$ over a small patch of surface $\mathrm{d}A$ located at x:

$$E(x) = \frac{\mathrm{d}^2\, Q(x)}{\mathrm{d}A(x)\,\mathrm{d}t}\,.$$

As we do not describe frequency-dependent effects, we omit the dependency on ν when expressing radiometric quantities. However, we make no assumption on the inner representation of spectra: triplets in a color space such as RGB, parameterized curves, *etc.*

Radiance The numerical methods on which our work is based estimate illumination by building light paths throughout the scene, and measuring their contribution on the sensor. To quantify the energy carried along paths we need to define radiance, which represents the irradiance traversing a surface element along a given direction ω:

$$L(x,\omega) = \frac{\mathrm{d}E(x,\omega)}{|\omega \cdot \mathrm{n}(x)|\,\mathrm{d}\omega}\,.$$

In this equation, $\mathrm{n}(x)$ is the surface normal at x and $\mathrm{d}\omega$ the infinitesimal solid angle, a small cone of directions around ω. The unit of radiance is $\mathrm{W.m^{-2}.sr^{-1}}$.

3.1.2 • The rendering equation

Under our assumptions, the propagation of light is influenced only by surfaces of the scene. To model this influence, we rely on the Bidirectional Scattering Distribution Function (BSDF). The BSDF ρ relates an incoming radiance field L_i with the corresponding outgoing radiance field L_o at point x of a surface, for any pair of directions ω_i, ω_o:

$$\rho(x,\omega_i,\omega_o) = \frac{\mathrm{d}L_o(x,\omega_o)}{L_i(x,\omega_i)|\mathrm{n}(x)\cdot\omega_i|\,\mathrm{d}\omega_i}\,.$$

Numerous BSDF models have been established to reproduce the behavior of real surfaces from a few parameters. Most of them use a *roughness* value α ranging from 0 to 1 to control the sharpness of the distribution. When α is close to 0, the BSDF has a narrow support over the sphere for a fixed incoming direction, and will reflect light like a mirror or refract it like glass. When α is close to 1, the support is large and scattering is smooth, distributing photons evenly across outgoing directions.

While the BSDF describes the variation of outgoing radiance relative to incoming radiance, a surface can also spontaneously emit light. The rendering equation [Kaj86] summarizes the two: emitted radiance is denoted L_e, and light scattering is expressed as the product of BSDF and incoming radiance:

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{\mathcal{S}^2} \rho(x,\omega_i,\omega_o)\,|\mathrm{n}(x)\cdot\omega_i|\,L_i(x,\omega_i)\,\mathrm{d}\omega_i\,. \tag{3.1}$$

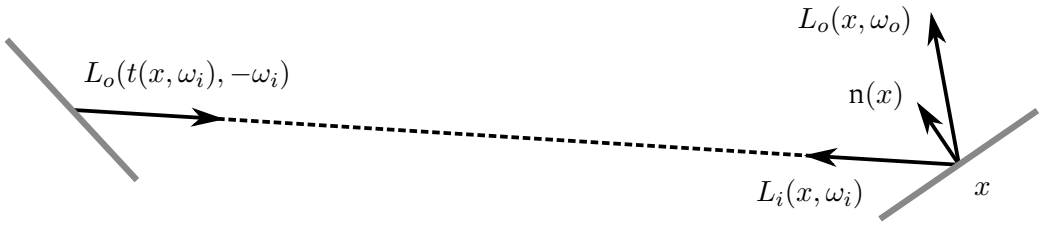


Figure 3.3: *The geometric setup of the rendering equation and ray-casting operator.*

The rendering equation thus describes a local energy balance. To describe radiance globally, we must relate incoming and outgoing radiance across different points. The ray-casting operator $t(x, \omega)$ associates x with the closest visible surface point in direction ω. As light rays travel in vacuum, we have the equality illustrated in Figure 3.3:

$$L_i(x, \omega_i) = L_o(t(x, \omega_i), -\omega_i)\,. \tag{3.2}$$

The association of equations (3.1) and (3.2) results in a recursive relationship between L_i and L_o, characteristic of light transport. Intuitively, we expect global illumination rendering algorithms to unroll these equations point after point, and to simulate light paths in the process. The formalization of this intuition is addressed later, in Section 3.2.

3.1.3 • Surface impact

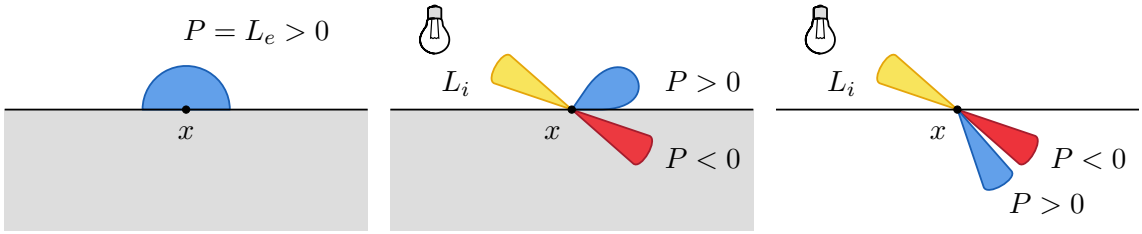
With the radiometric quantities and general rules of light propagation defined, we introduce *impact*, which represents the influence that surfaces have on the distribution of radiance in the scene. Impact is understood locally by introducing a new quantity P, the difference between outgoing and incident radiance:

$$P(x, \omega_o) = L_o(x, \omega_o) - L_i(x, -\omega_o)\,.$$

When there is no surface at x, the linearity of light transport implies that $P = 0$. However if point x is located on a surface, the relationship between L_i and L_o is known and given by the rendering equation (3.1). We develop the expression of P in this case:

$$\begin{aligned} P(x, \omega_o) &= L_e(x, \omega_o) + \int_{\mathcal{S}^2} \rho(x, \omega_i, \omega_o)\, |\mathrm{n}(x) \cdot \omega_i|\, L_i(x, \omega_i)\, \mathrm{d}\omega_i - L_i(x, -\omega_o) \\ &= L_e(x, \omega_o) + \int_{\mathcal{S}^2} \left(\rho(x, \omega_i, \omega_o) |\mathrm{n}(x) \cdot \omega_i| - \delta(\omega_i + \omega_o)\right) L_i(x, \omega_i)\, \mathrm{d}\omega_i\ . \end{aligned} \tag{3.3}$$

The Dirac distribution $\delta(\omega_i + \omega_o)$ is non-null only when $\omega_o = -\omega_i$, and integrates to 1. For a fixed surface point and outgoing direction, let alone the emitted radiance term L_e, we interpret impact as a measure of the difference between the actual reflectance and a perfect transmission over the sphere of incoming directions. Furthermore, Figures 3.4 and 3.5 show that while impact depends on the properties of the surface at x, its behavior is always twofold: a positive part is brought by scattering and emitted radiance, and a negative part is due to incoming light not passing through the surface.



Diffuse emissive surface *Glossy reflective surface* *Specular refractive interface*

Figure 3.4: *The sign of impact in the sphere of directions depends on the behavior of the surface at point x, which is described by the BSDF and the emitted radiance field.*

Following this first distinction, impact appears as a promising quantity to represent the amount of light gained and lost due to surfaces of the scene. From an artistic point of view, isolating the impact of certain surfaces of interest inside separate AOVs has the potential to simplify compositing tasks. Intuitively, the negative part of impact is related to the apparition of shadows in the image, and will be the subject of Chapters 4 and 5. The rest of this chapter is devoted to isolating the positive part of impact, which encompasses a broad range of lighting features such as reflections and highlights.

3.2 • Path integral formulation

To render images based on the physical rules of light transport, and isolate positive impact into separate AOVs, it is necessary to transform the local equations between radiometric quantities into a global rendering algorithm. This is precisely the goal of the path integral formulation, introduced in the seminal work of Veach [Vea97].

3.2.1 • Definition

In the path integral framework, the measure I_j of radiance at sensor position j is expressed as an integral over Ω the space of all light paths in the scene:

$$I_j = \int_\Omega f_j(\bar{x}) \, \mathrm{d}\mu(\bar{x}) \, . \tag{3.4}$$

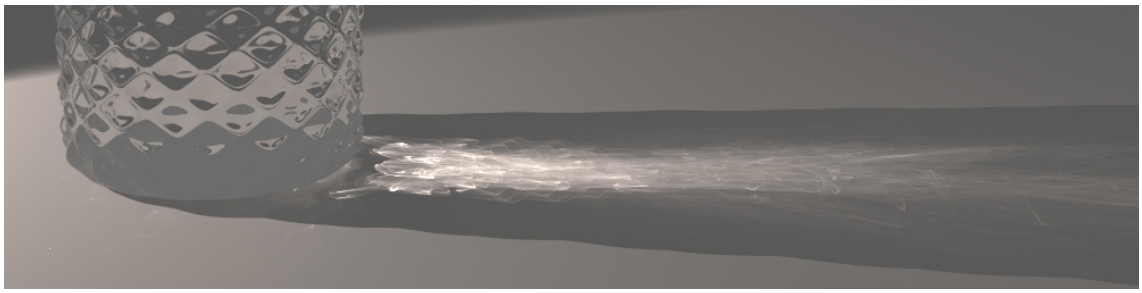


Figure 3.5: *The surface of the glass slab projects a mixture of caustics (positive impact) and shadow (negative impact) onto the ground plane.*

Based on this equation, Monte Carlo integration is applied to create estimators that approach the measure of radiance on the sensor by generating N random light paths. The main advantage of Monte Carlo methods is that the estimation error converges in $\mathcal{O}(N^{-1/2})$, regardless of the number of dimensions in the integrand.

The path space

Our interest in this formulation is also motivated by the presence of the path space Ω, which has an artistic value besides its technical purpose. Ω is the space of all geometric paths that can be constructed in the scene. A path $\bar{x}$ of length k is defined by the position of its $k+1$ successive vertices on $\mathcal{M}$, the union of surfaces that compose the scene:

$$\bar{x} = \mathrm{x}_0\,\mathrm{x}_1\,\ldots\,\mathrm{x}_k \quad \text{where} \quad \forall i,\, \mathrm{x}_i \in \mathcal{M} \subset \mathbb{R}^3\,.$$

To account for paths of any length, Veach first defines set Ω_k that contains the paths of length k. Then, Ω is obtained by considering the countable union of all Ω_k:

$$\Omega = \bigcup_{k=1}^{+\infty} \Omega_k\,.$$

Elements of the path space do not always correspond to real light paths formed by the propagation of photons emitted from a light source, as physically impossible light paths exist in Ω. It is the role of the measure μ and the measurement contribution function f to determine respectively the weight of a path, and the energy it transports.

Path measure

The measure μ associates a path $\bar{x}$ with a certain probability density $\mathrm{d}\mu(\bar{x})$ given by the product of the area measure A on surfaces:

$$\mathrm{d}\mu(\bar{x}) = \mathrm{d}\mu(\mathrm{x}_0\,\mathrm{x}_1\,\ldots\,\mathrm{x}_k) = \mathrm{d}A(\mathrm{x}_0) \times \ldots \times \mathrm{d}A(\mathrm{x}_k)\,.$$

Then at each vertex, $\mathrm{d}A$ develops into the usual area measure. For example, $\mathrm{d}A = \mathrm{d}x\,\mathrm{d}y$ if a local Cartesian parameterization of the surface exists at the vertex location.

Measurement contribution function

The last step in the definition of the path integral formulation is that of the measurement contribution function f, wherein reside the physical aspects encountered in Section 3.1. Function f associates a light path with the energy it carries from a light source to the sensor location j, and results from the recursive expansion of the rendering equation (3.1):

$$\begin{aligned} f_j(\bar{x}) = {} & L_e(\mathrm{x}_0, \mathrm{x}_1) W_e^{(j)}(\mathrm{x}_{k-1}, \mathrm{x}_k) G(\mathrm{x}_0, \mathrm{x}_1) V(\mathrm{x}_0, \mathrm{x}_1) \\ & \cdot \prod_{i=1}^{k-1} \rho(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1}) G(\mathrm{x}_i, \mathrm{x}_{i+1}) V(\mathrm{x}_i, \mathrm{x}_{i+1})\,. \end{aligned} \tag{3.5}$$

$L_e(\mathrm{x}_0, \mathrm{x}_1)$ is the radiance emitted from x_0 toward x_1, and $W_e^{(j)}(\mathrm{x}_{k-1}, \mathrm{x}_k)$ the importance leaving the sensor from x_k toward x_{k-1}. Importance is the adjoint quantity of radiance, and is emitted by the sensor: it models its sensibility to incoming light rays.

Compared to Section 3.1.2, the BSDF is now parameterized by the three vertices x_{i-1}, x_i, and x_{i+1}. This notation can be expressed based on the standard form:

$$\rho(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1}) := \rho\left(\mathrm{x}_i, \frac{\mathrm{x}_{i-1} - \mathrm{x}_i}{\|\mathrm{x}_{i-1} - \mathrm{x}_i\|}, \frac{\mathrm{x}_{i+1} - \mathrm{x}_i}{\|\mathrm{x}_{i+1} - \mathrm{x}_i\|}\right).$$

Lastly, the geometric factor G is defined as

$$G(x, y) = \frac{|\cos(\theta_x)\cos(\theta_y)|}{\|x - y\|^2},$$

where θ_x and θ_y are the angles between the segment xy and the surface normals at x and y respectively. The visibility $V(x, y)$ is 1 if x and y are mutually visible, and 0 otherwise.

3.2.2 • Estimation

The base principle of Monte Carlo integration is to consider the integral I_j as the expected value of a random variable. Suppose that X is a random variable modeling light paths, which follows a probability density function $p(X)$. If $p(X) \neq 0$ when $f_j(X) \neq 0$, then the expected value of $f_j(X)/p(X)$ under the measure $\mathrm{d}\mu$ is

$$\mathbb{E}\left[\frac{f_j(X)}{p(X)}\right] = \int_\Omega \frac{f_j(\bar{x})}{p(\bar{x})} p(\bar{x})\,\mathrm{d}\mu(\bar{x}) = \int_\Omega f_j(\bar{x})\,\mathrm{d}\mu(\bar{x}) = I_j\,.$$

Given a suitable p, an estimator is used to approximate the expected value. Typically, the empirical mean is taken over N independent realizations of X:

$$\mathbb{E}\left[\frac{f_j(X)}{p(X)}\right] = \lim_{N \to +\infty} \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}\,. \tag{3.6}$$

The probability density function is responsible for the variance of the estimator, and thus the amount of noise in the image. More precisely, that variance is null when p and f are proportional [Vea97]. While this is not achievable in practice, Monte Carlo rendering algorithms, or *integrators*, strive to sample the path space efficiently by generating random paths according to a probability density function as close as possible to f.

Path tracing

Path tracing [Kaj86] is an integrator that builds light paths incrementally starting from the sensor, and searching for an intersection with the scene. At each new vertex, two operations are performed.

- First, the algorithm samples direct illumination from light sources in the scene. If a non-null contribution is found, a path is formed that connects the sensor with a light source; its contribution is accumulated in the empirical mean of Equation (3.6). Known as *next-event estimation*, this optimization greatly reduces variance by ensuring a consistent number of contributions.
- Then, in order to properly sample the integrand f, the next direction of propagation is sampled according to the BSDF at the current location.

This iterative procedure stops after a fixed path length to avoid an infinite recursion; advanced termination criteria are discussed in Section 4.6.1. Because of its simplicity and robustness, path tracing is widely used in production [Fas19].

3.3 • Light path expressions

One information we retain from the previous section is that Monte Carlo integrators all explore the high-dimensional space Ω by generating numerous random light paths. This common characteristic has already been used as an entry point for artistic editing in previous work (Section 2.4.2), but the main liability of the existing methods was the expensive re-rendering of the scene needed after each edit. Only light path expressions used information from global illumination while proposing real-time editing.

Connection with positive impact

We have seen in Section 3.1.3 that positive impact is created when a surface of the scene emits or scatters light toward an outgoing direction. Monte Carlo integrators are fully aware of the occurrence of these events at vertices of the light paths they sample, and can be leveraged to extract impact during the simulation. LPEs emerged from this observation, and have become an important tool of the compositing pipeline in production [Bre20].

Their base principle is to describe clusters in the path space using regular expressions that match the vertices of light paths. Each cluster describes a lighting feature such as a reflection or a caustic, based on the propagation history of the paths it contains. The rendering algorithm outputs the contribution of clusters inside separate AOVs, isolating their impact in image space. According to Equation (3.4), these AOVs will have a linear relationship: integration turns a disjoint union of path space clusters into a sum. In turn, image space compositing edits are based on linear operators, and quick to process.

Heckbert notation

To form clusters inside the path space with regular expressions, we must use a set of symbols to match vertices along light paths. Heckbert proposed a taxonomy based on four event types defined by letters [Hec90] to describe the different events that rule propagation: light emission (**`E`**), scattering (**`D`** and **`S`**), and reception on the sensor (**`E`**). This notation is combined with the features of regular expressions such as the wildcard **`.`** to match any symbol, or quantifiers that represent multiple symbols at once: **`*`** or **`+`** for instance.

Variations on Heckbert notation have been proposed to further extend its possibilities, such as Veach's full-path regular expressions [Vea97] that better describe the properties of light sources and sensors. However, Heckbert notation was originally meant to classify rendering algorithms according to the type of paths they can generate; its final purpose was to combine the strengths of multiple algorithms to obtain all possible types of lights paths efficiently. De facto, the meaning of the four original symbols is purely theoretical and does not allow much flexibility when clustering light paths according to an artistic goal. Based on this observation and on the knowledge of production requirements, the Academic Software Foundation has since designed a wider set of symbols and rules.

3.3.1 • Open Shading Language

The Open Shading Language specification (OSL, [Gri16]) contains a thorough set of rules to create finer clusters of light paths. The OSL defines *event types* to describe the location of interactions along paths:

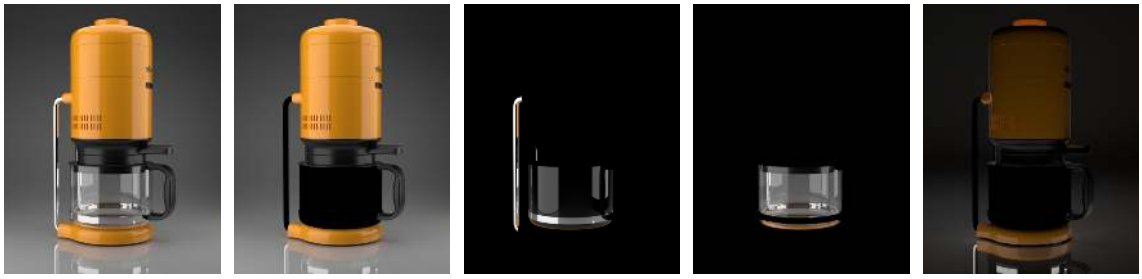
- **C** Refers to the camera endpoint.
- **L** Identifies a light source.
- **O** Denotes emissive objects; the main difference with **L** is that these objects do not undergo next-event estimation: their emission is accumulated only when the path encounters them during propagation.
- **R** Is for surface reflections, where the incoming and outgoing direction are located in the same hemisphere relative to the surface normal.
- **T** Is for surface transmissions, if the hemispheres are opposites.
- **V** In the case of volume scattering inside a participating medium.
- **B** When the ray does not intersect the scene and ends up in the background, at infinity.

If the event type corresponds to a scattering interaction, then an *event scattering type* can be added after it inside angle brackets to refine the selection:

- **D** Corresponds to diffuse scattering, with a finite BSDF over the sphere of directions.
- **G** Is reserved for glossy BSDFs that are modeled with a roughness parameter α.
- **S** Denotes perfectly specular scattering by an infinite BSDF; essentially, combinations of reflecting or transmitting Dirac distributions.
- **s** For a straight transmission, when the BSDF is similar to the distribution $\delta(\omega_i + \omega_o)$ that we introduced to characterize impact in Section 3.1.3.

Thanks to this disambiguation, a diffuse reflection is matched by **<RD>**, and a specular transmission by **<TS>**. The direction of expressions matches that of path tracing, and describes paths from the camera to the light sources. Several examples of light path expression based on scattering types are shown in Figure 3.6.

Another addition of OSL is the support for string identifiers in light path expressions. In production software, all entities are labelled with unique strings that can be added in last position between the angle brackets. For example, **.*<L.'key_light'>** identifies the light paths that carry a contribution from the key light to the camera.



Beauty **.*** *Glossy* **CG.*** **C<RS>.*** **C<TS>.*** **C<RG><R.>L**

Figure 3.6: *Light path expressions allow the separation of impact according to the type of scattering interaction inside separate layers, here in the Coffee Maker scene.*

```
WorldBegin
    LightSource "distant"
    Material "plastic"
    Shape "sphere"
WorldEnd
```

Code 3.1: *Basic syntax of a pbrt scene.*

```
AttributeBegin
    Identifier "point_light"
    LightSource "point"
AttributeEnd
```

Code 3.2: *Adding an identifier.*

```
FilmBegin "layered"
    # Beauty AOV
    Film "image"
        "string filename" ""
    # LPE AOVs
    Film "lightpath"
        "string filename" "Glossy"
        "string expression" "CG.*"
    Film "lightpath"
        "string filename" "Specular"
        "string expression" "CS.*"
FilmEnd
```

Code 3.3: *Specifying multiple AOVs to export.*

3.3.2 • Implementation

We have implemented several features from the OSL specification to work with light path expressions in the open source, physically-based renderer pbrt-v3 [Pha16].

Scene description

The pbrt executable is not associated with a graphical user interface; instead, it is invoked from the command line to render scenes described by structured text files. The syntax is based on scopes enclosed by `*Begin` and `*End` keywords. Inside each scope, instructions are read and applied from top to bottom as seen in Code 3.1.

Moreover, pbrt does not support the identification of scene entities by default. We thus add the possibility to name objects using the `Identifier` keyword in the scene description (Code 3.2). Its only parameter is a user-provided string that applies to light sources and shapes located below in the scope, and is retrieved at runtime.

Normally, pbrt outputs a single image corresponding to a `Film` entity described in the scene. In our implementation, a scene file can contain any number of films to export multiple AOVs during rendering. If a multi-layered EXR image is preferred to a collection of images, a special scope must be opened and populated with several AOVs that each create a layer in the image, as demonstrated in Code 3.3.

Language automata

An efficient implementation to match regular expressions based on language automata is provided in open source by the Academic Software foundation. We proceed as follows: for each light path expression, an automaton that recognizes it is created. When a new path tracing sample begins, all automata are reset to their initial state. Each new vertex along the path is submitted to the automata, and may trigger a transition (Figure 3.7).

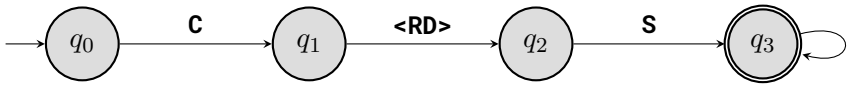


Figure 3.7: *This language automaton matches* ***C<RD>S.****, *light paths that create caustics. It is adapted to path tracing, as vertices are sampled in the order the automaton treats them.*

Whenever a contribution is gathered, it is recorded in all the images where the automaton is in a final state. This procedure proceeds naturally with path tracing, as vertices are sampled in the same order the LPEs describe them.

Extension to others AOVs

Positive impact is an important quantity to obtain in image space, but we also enable the export of other information from the first vertex of each path inside separate layers.

Depth Contains the distance between the vertex and the starting point at the sensor. It is often used to mimic atmospheric attenuation effects, or to perform compositing operations in a 2.5D space by un-projecting pixel locations.

Normal Records the normal of the surface in world space coordinates, and is leveraged for relighting if the surface receives illumination from a known direction.

Albedo Exports the value of the hemispherical-directional reflectance at ω_o, the outgoing direction of the ray. This quantity is defined as $\int_{\mathcal{H}^2} \rho(x, \omega_i, \omega_o) |\omega_i \cdot \mathrm{n}(x)| \, \mathrm{d}\omega_i$ and is computed in closed form for some BSDFs. Otherwise, the program integrates it numerically based on a user-controlled number of direction samples.

Mask Takes as input an identifier, and outputs a value of 1 when the corresponding entity is encountered, and 0 everywhere else. This AOV is also called a *matte* in compositing, and used to control the extent of filters or the mix factor between two layers.

These AOVs are akin to geometric buffers or *G-buffers*, which clarifies their importance for post-processing. They contain the base quantities used in deferred shading [Dee88], meaning that they carry most of the information necessary to render direct illumination in image space; this explains their use as utility layers for relighting. In their original definition [Sai90], G-buffers were also used to enhance the shape of objects by outlining their edges, or to automate stylization by shading objects with hatches.

3.3.3 • Results and applications

To illustrate the use of LPEs and utility AOVs in compositing, we present various edits on two different scenes rendered with our pbrt-v3 implementation.

The Coffee Maker scene displays sharp glossy reflections on the body of the machine, and contains specular surfaces on the metal bar and the pot. Given the complexity of light transport in such a setup, a common task in compositing is to explore the lighting design space in real-time by adding many potential light sources into the scene and separating their contribution in the image. In Figure 3.8, we show that the combination of identifiers and LPEs allows such a separation. From this set of layers, a compositing artist can modulate the influence of each source in isolation thanks to their linear relationship, and assess what combination of light intensities better approaches the intended result.

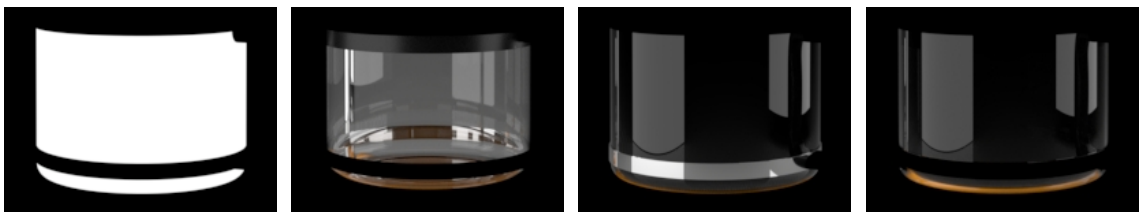


`.*<L.'key'>` `.*<L.'fill'>` `.*<L.'top'>`

Figure 3.8: *Identifiers work for light sources, and allow the separation of their contribution.*



(a) *Beauty* (b) *Blur and coffee* (c) *Blurred* `C<RG'floor'>.*`



(d) *Mask of the pot* (e) `C<TS><TS>.*` (f) `C<RS'pot'>.*` (g) `C<TS><RS'pot'>.*`

Figure 3.9: *The use of object identifiers inside light path expressions enables advanced edits. Top row: compared to the original (beauty) render, the result also displays blurred reflections on the floor. Bottom row: the various AOVs used to composite coffee inside the pot.*

Advanced edits are presented in Figure 3.9. In comparison to the original or *beauty* render (Figure 3.9a), the composite result (Figure 3.9b) has two main differences. Using a first LPE, we isolate and blur the reflections on the floor (Figure 3.9c). Second, we composite coffee inside the pot using the AOVs shown in the bottom row. Based on a mask of the pot (Figure 3.9d), we create a brown shape to represent the liquid. The decomposition of lighting using LPEs allows us to respect the propagation of rays inside (Figure 3.9e) and on the surface (Figures 3.9f and 3.9g) of the glass pot.

Figure 3.10 shows the composite result corresponding to the graph of Figure 3.1 in the top row, based on the Marbles scene. In the middle row, we display some of the LPEs that are used in the process. They allow a precise extraction of advanced global illumination effects such as the caustics cast by the glass sphere, and the radiance it transmits toward the camera. These lighting features are then color-graded separately to completely transform the look of the scene. On the mirror sphere in the center, we used a non-linear *glow* filter to emphasize the two light sources that are seen in reflection. The bottom row contains three utility layers: the depth, normal, and albedo AOVs exported by our implementation from first-bounce information.

3.4 • Limitations of LPEs in compositing

The beginning of this chapter has served as an introduction to the physics of light transport, on which are based realistic rendering algorithms. We have seen that the notion of impact characterizes the influence of a surface on the radiance field; its positive part accounts for the apparition of lighting features such as reflections or caustics, and is conveniently taken into account by the path integral formulation. Describing clusters inside the path space using light path expressions allowed us to precisely isolate part of the positive inside separate layers, and we demonstrated their use in compositing.

However, the negative part of impact remains elusive as it is not explicitly carried by light paths, and thus out of reach for traditional Monte Carlo integration and light path expressions altogether. The next chapters explore negative impact in depth, and uncover its connection with shadows.

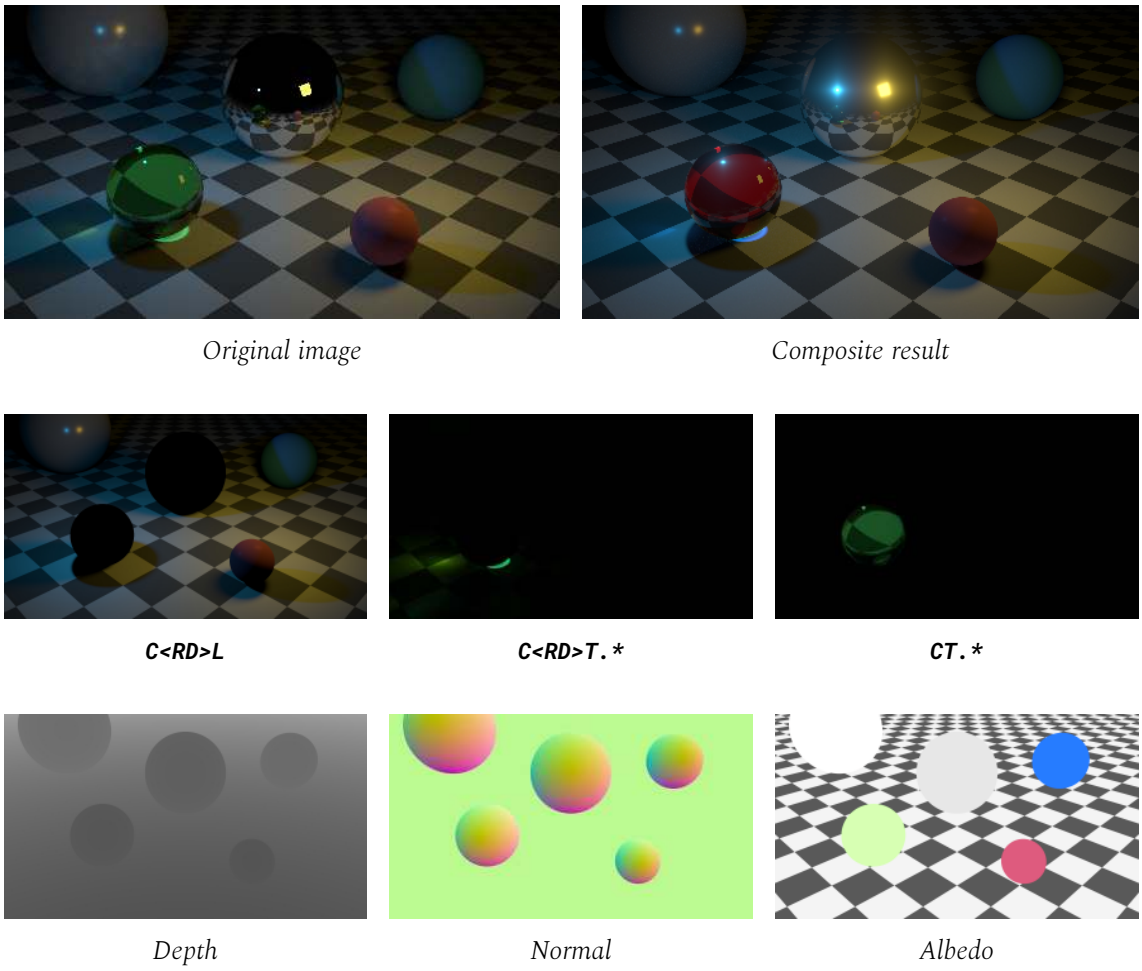


Figure 3.10: *Top row: the compositing graph of Figure 3.1 uses LPEs in the Marbles scene (middle row). It changes the appearance of various lighting features such as the caustic and transmissions of the glass sphere. Bottom row: the utility AOVs exported by our implementation.*

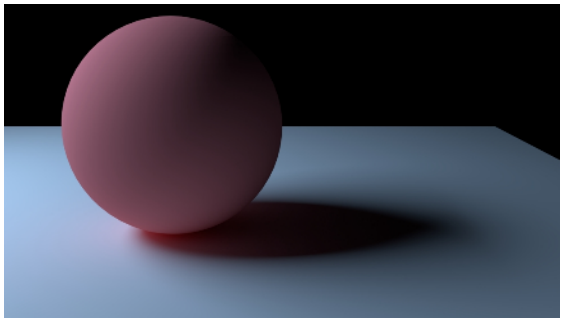
Chapter 4

Shadow Layers for Solid Objects

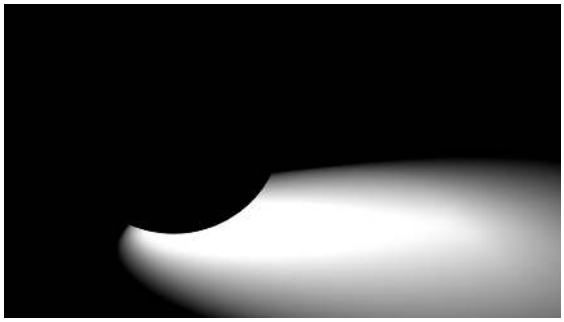
Light path expressions are a powerful tool to transform the appearance of lighting features in compositing, but the theoretical context introduced in Chapter 3 has also shown that they do not apply to shadows, that are never explicitly simulated by physically realistic rendering methods. Nevertheless, many available renderers propose options to help in the compositing of shadows, and we review them in Section 4.1. We highlight two issues in existing software: while extracted from global illumination, all available information on shadows is limited to primary light sources; we also show that three well-established renderers overestimate illumination when shadow is disabled for an object.

Our goal in this chapter is to design AOVs for shadow compositing that overcome the limitations underlined above. We propose to characterize shadow as the negative part of light impact, which leads us to a first definition of *shadow layers* in Section 4.2. A shadow layer contains the radiance lost on the sensor due to a single solid object, and is obtained by image space subtraction between two specific renders of the scene. This definition is directly supported by any rendering software, but has several limitations such as its computational complexity, requiring two additional renders per object of interest.

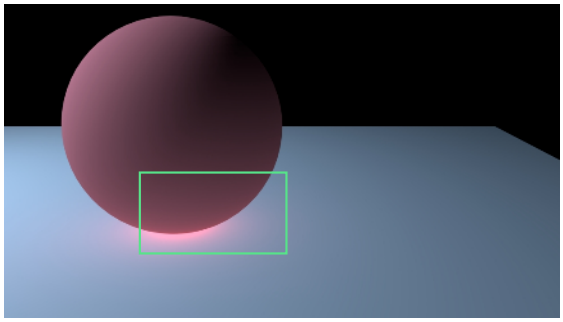
Seeking an efficient solution, we express the subtraction operation directly in the path space, which leads us to a path integral formulation for shadow presented in Section 4.3. This definition is the starting point for Monte Carlo estimation of shadow layers; Section 4.4 presents our modified version of the path tracing algorithm that renders the original image and any number of shadow layers associated with different solid objects in a single pass. We show how to integrate user control in the rendering process to refine the appearance of shadow layers from a few parameters. Section 4.5 then discusses the implementation of our pbrt-v3 prototype in terms of rendering time and image convergence on several scenes that exhibit challenging light transport. For each scene, we include post-processing edits that demonstrate the convenience of shadow layers in compositing. Our method is amenable to Monte Carlo algorithms other than path tracing: Section 4.6 gives implementation guidelines for the most popular integration schemes. Because our method explicitly builds paths that carry shadow, we can now leverage light path expressions to render shadow layers; we present an Arnold for Maya plugin prototype based on this approach.



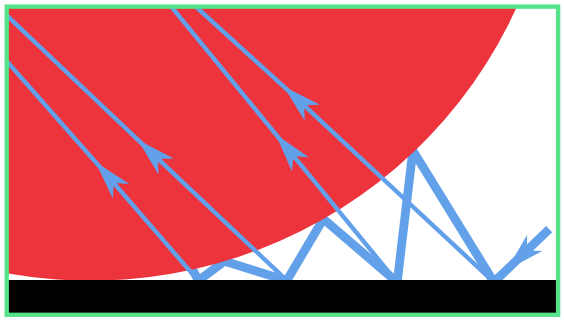
(a) *Original render*



(b) *Shadow matte generated by Arnold*



(c) *pbrt, Cycles, Arnold: shadow disabled*



(d) *Long path near the contact point*

Figure 4.1: *The Simple Sphere scene serves as a baseline to assess the capabilities of conventional renderers. It outlines a common issue they encounter when disabling shadow: radiance is overestimated along light paths that encounter many occlusions, as detailed in (d).*

4.1 • Limitations of existing software

Many global illumination renderers already provide ways to manipulate shadows in a scene. We have found that a common functionality in available software is to disable shadows on a per-object basis. Using this option, the renders with and without shadows are combined in compositing to manipulate the appearance of shadows. We focus on three well-established programs that propose to disable shadows:

- pbrt-v3, path tracer, setting the shape parameter ”shadowalpha” to 0.
- Blender 2.78 Cycles, disabling the object option ”Shadow”.
- Arnold 5.2.2 for Maya, toggling off the shape option ”Cast Shadows”.

We use a simple scene to compare these three different renderers. The scene is composed of two Lambertian surfaces, a red sphere on a white plane, exposed to a blue area light source located in the top left corner (Figure 4.1a).

Shadow overestimation

In Figure 4.1c, we see that the three renderers give the same result when shadows are toggled off. In the vicinity of the contact point between the sphere and the plane, a strong red reflection appears that cannot be explained knowing the position of the light source. While the left part of the reflection is legitimate, the portion enclosed by the green rectangle is visibly an artifact that all three renderers were proven to exhibit.

We uncovered the reason for this overestimation from the source code of pbrt-v3. The images are generated by path tracing, and at every vertex of the path the algorithm estimates direct lighting from the source. If the surface on which the vertex lies is oriented toward the source, a shadow ray is cast to check its visibility. With shadow disabled, this visibility check always passes. As demonstrated in Figure 4.1d, this means that long paths near the contact point between the plane and sphere bring a near-constant contribution of radiance at every other bounce, the surfaces being purely diffuse.

Indirect shadows

Another option proposed by a number of renderers is the *shadow matte*, a grayscale layer representing the amount of shadow in the scene. This shadow matte is used as an input to control the parameters of image space filters, or to re-create the desired appearance from a shadowless version. An example of shadow matte generated with Arnold is shown in Figure 4.1b; it contains the ratio of non-occluded shadow rays at each pixel [Aut21b].

The main issue with conventional shadow mattes is that they derive from the estimation of direct illumination only. As a result they do not account for indirect shadows, created by the occlusion of secondary light sources. This shortcoming is emphasized by Figure 4.2, where lighting is almost entirely indirect. In this setup, the secondary light source is the surface of the umbrella that reflects light incoming from the projector. Arnold does not detect the occlusion of this secondary source by the dragon, and thus fails to pick up indirect shadow over the pedestal.

These two shortcomings of existing renderers are known by lighting and compositing artists [Lle19], who have to correct them by hand. The method we propose shares the same technical context as existing software, but provides an accurate estimation of direct and indirect shadow, solving both issues.

4.1.1 • Overview of our method

Our objective is akin to that of the shadow matte: we wish to render additional shadow layers that contain the radiance lost on the image because of the presence of an object in the scene. Contrarily to the shadow matte, a shadow layer does not simply indicate the presence of shadow, but has an intrinsic physical meaning. Intuitively, shadow layers are related to the negative part of impact that was discussed in Section 3.1.3.

Much like standard light path expressions account for the positive part of impact in image space, shadow layers are designed to recover its negative part as it accounts for the incident light that is diverted by the surface. This connects with our intuitive definition of shadow as a local loss of energy due to the occlusion of rays. The main difference with positive impact is that shadow is not carried by light paths, and thus not considered during Monte Carlo rendering; it is a natural product of the simulation. Nevertheless, our goal is to design a rendering algorithm that allows the extraction of shadow inside separate layers. To do so, the next section presents the different entities involved in the creation of shadow and how we identify them.

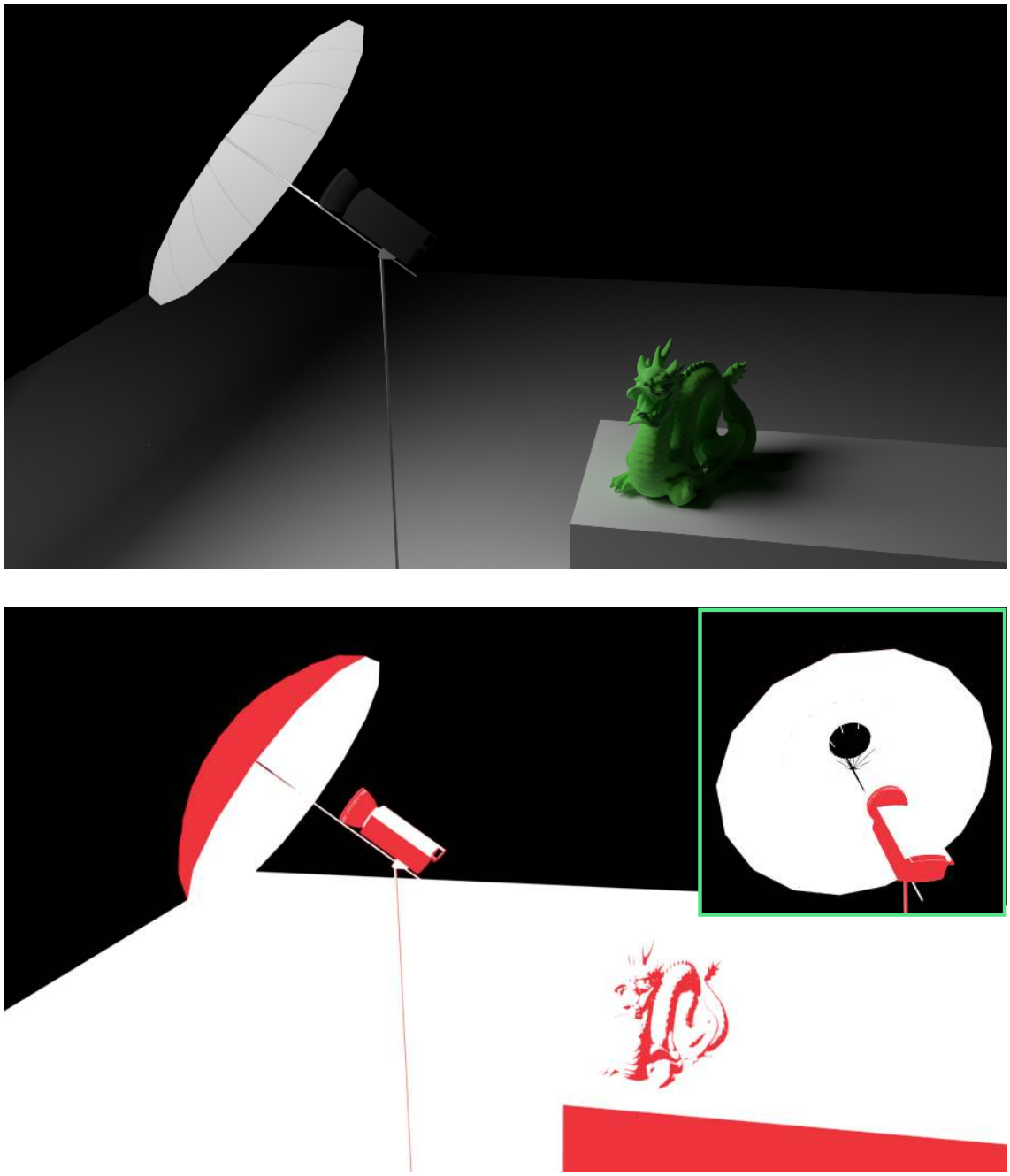


Figure 4.2: *Top: in the Dragon scene, illumination is indirect as the light of the projector is reflected before reaching the subject. Bottom: shadows should be indicated in white; Arnold fails to pick up indirect shadows in this setup. We colored backfacing surfaces in red for clarity, but they originally appear black and are considered not in shadow. In inset, only the area near the apex of the umbrella is correctly marked. The rest of the scene is considered to be in shadow.*

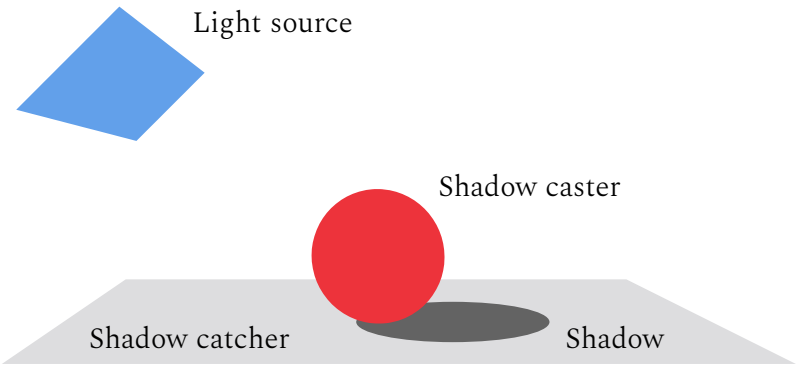


Figure 4.3: *The chosen naming convention involves three objects.*

4.1.2 • Involving objects

We consider that the scene is only composed of solid objects, and the union of their surface is denoted $\mathcal{M}$. We settle a permissive definition for an object $\mathbf{o}$, and identify it with the surface $\mathcal{M}_\mathbf{o} \in \mathcal{M}$ it spans inside the scene. No further assumptions are made on $\mathcal{M}_\mathbf{o}$: it is not required to be manifold, and its topology can be disconnected. Based on this definition of objects, Figure 4.3 illustrates the naming convention we detail below.

Light source Shadows result from the occlusion of illumination coming from a light source, that can be of any type: a point light, an area light, *etc.* This source emits a light field L_e; along its propagation, L_e is at some point diverted from its original destination by an object that scatters or absorbs incoming light.

Shadow caster The object that diverts the incoming light field and creates shadow is called the *shadow caster*. This consolidates our analysis of impact: we interpret the shadow caster as emitting a quantity complementary to radiance, that generates negative impact. This quantity has been called *antiradiance* in the literature [Dac07].

Shadow catcher A *shadow catcher* is an object that displays shadow on its surface. When the shadow caster and catcher are the same object, the result is *self-shadowing*; otherwise, the corresponding lighting feature is a *cast shadow* (Figure 4.4).

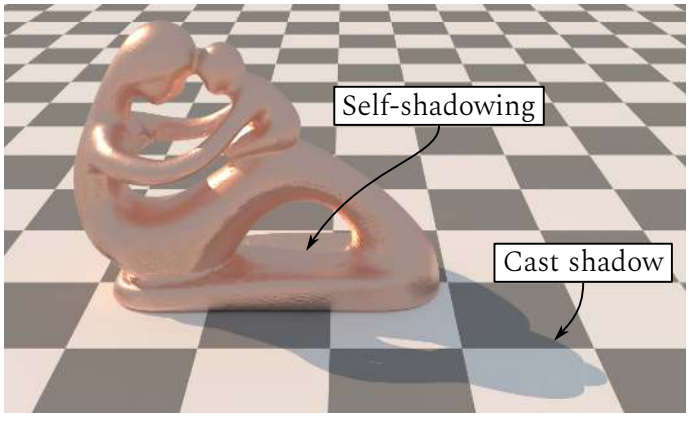


Figure 4.4: *Depending on the catcher, self-shadowing or cast shadows appear.*

4.2 • Shadow from image subtraction

Based on the notion of impact presented in Section 3.1.3, we present a method to obtain the shadow layer of a shadow caster **c** based on an image space subtraction.

4.2.1 • The intuition

We have outlined that shadow corresponds to the negative part of impact, where emitted radiance L_e is ignored as it only brings positive contributions. In that case, impact P quantifies the difference in outgoing radiance between a surface described by $\rho(x, \omega_i, \omega_o)$ and an invisible surface. After overriding the BSDF of our shadow caster of interest **c**, we render two specific versions of the scene that allow us to obtain its impact in image space.

Invisible version By overriding the BSDF to $\delta(\omega_i + \omega_o)/|\mathrm{n}(x) \cdot \omega_i|$, we obtain a straight transmitting surface over **c**. In terms of impact, Equation (3.3) states that $P = 0$ over the caster. Rendering this version of the scene yields image J where the impact of **c** on the light field is null. In practice, this is achieved by simply removing **c** from the scene.

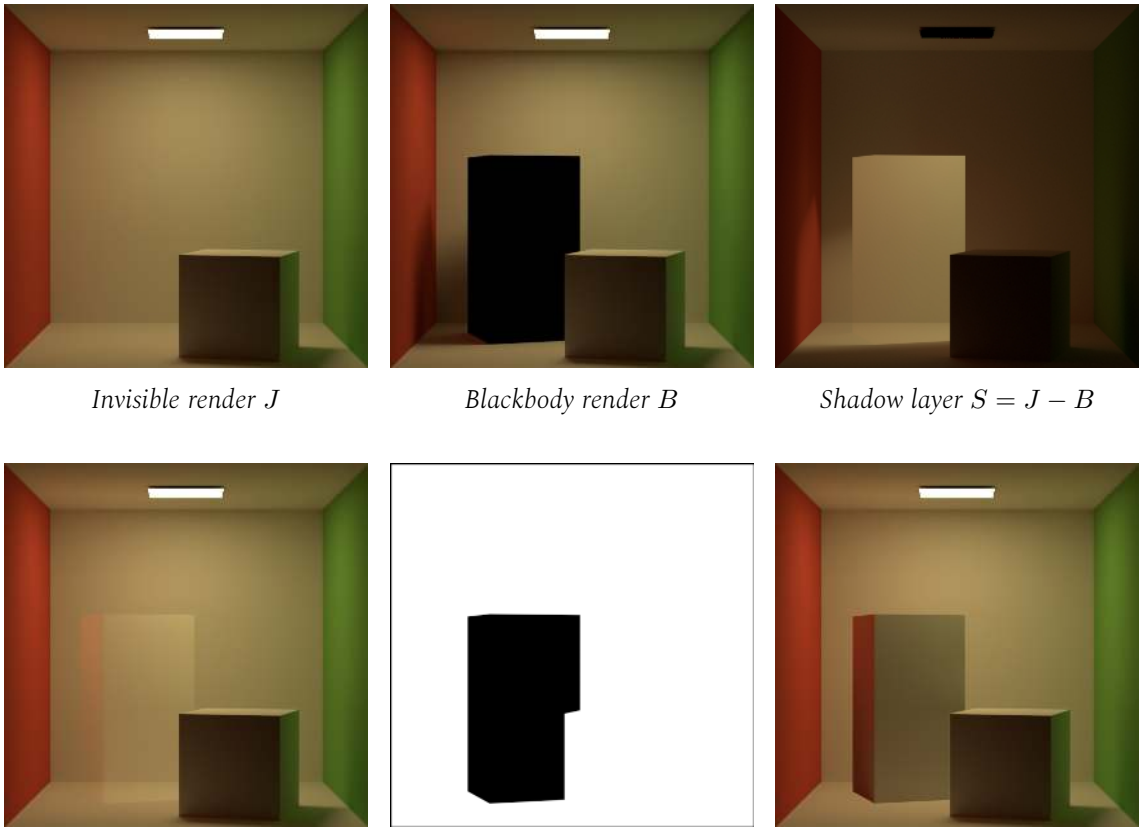
Blackbody version If we set $\rho(x, \omega_i, \omega_o) = 0$, caster **c** becomes a pure absorber. It follows that $P = -L_i$ at any point on the surface of **c**, where L_i is the incident radiance field. The corresponding image B determines the negative impact that caster **c** has on the rest of the scene, as it absorbs all incoming light without re-emitting any. We call this version the *blackbody*, as the physical realization of this behavior is a blackbody at temperature $0°\mathrm{K}$. However, any rendering program can simulate it using a black Lambertian surface.

The next property we use is the linearity of light, which transfers to image space as long as we use a linear encoding of pixel colors. When computing $J - B$, the impact that is measured on the sensor corresponds to $0 - (-L_i) = L_i$: we have recovered the amount of incident radiance that is occluded by **c**. This implies that the shadow layer contains the radiance missing due to the presence of the caster, and has a linear relationship with the original image: the addition of the two removes shadows created by **c**.

4.2.2 • Examples

The subtraction method is applied to the Cornell Box in Figure 4.5, where the shadow caster of interest is the tall box in the back. After the subtraction, shadow is removed from the original image by addition and disappears from the ground. However, the background wall is seen leaking where the box is directly visible from the camera and severely alters the appearance of the surface. In a compositing context, this is solved by creating a mask M of the object, and using it to limit the influence of S in the result.

The reason why the background wall is leaking when we remove shadow is that the shadow caster, the object on which missing energy is measured, is the camera sensor itself. If the sensor takes the shape of a rectangle in the scene, the subtraction method measures the amount of radiance that is missing from this rectangle due to the presence of the tall box. The leaking becomes understandable, as many rays between the background wall and the camera sensor are occluded by the box. Yet, our intuition of shadow is different from this behavior: we expect it to be measured on surfaces of the scene instead.



Invisible render J *Blackbody render B* *Shadow layer $S = J - B$*

Shadow removal via $I + S$ *Mask M* *Removal with mask $I + M \times S$*

Figure 4.5: *In the Cornell Box, the chosen caster* **c** *is the tall box. Its shadow layer S is computed as $J - B$. The captured shadow is removed from the original render by $I + S$. However, the background wall is seen leaking; controlling the influence of S with a mask M solves the issue.*

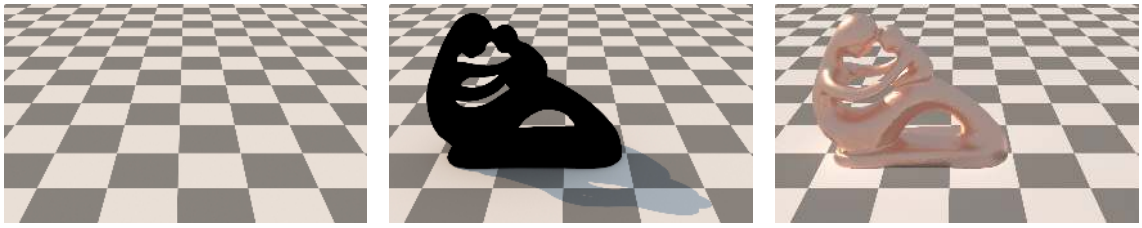
This shortcoming is particularly problematic if the caster exhibits self-shadowing on its surface, as seen in Figure 4.6. Because shadow is picked up on the camera and not on the surface of the statue, the method does not allow the extraction of self-shadows.

4.2.3 • Discussion

This approach gives us some insights on the formation of shadows, that correspond to the difference between an object with no impact and an object with only negative impact. One of its strength is that it can be used with any rendering software, as the blackbody and invisible versions of the scene are straightforward to obtain.

Additionally, image subtraction is also relevant to obtain the positive component of impact via $I - B$, as the corresponding result on surfaces is $P - L_i$. However, we have seen that such an image is already available using a light path expression of the form `.*<[RT].'caster'>.*`. The latter is preferable, as the estimation errors are cumulative when performing the subtraction and become less noticeable using the LPE.

A limitation of this approach for shadow is that the catcher being considered is invariably the camera sensor, which leads to artifacts where the shadow caster is directly visible.



Invisible render J *Blackbody render B* *Shadow removal using a mask*

Figure 4.6: *The Fertility Statue underlines the issue with direct visibility in the subtraction approach: as we use a mask on the shadow layer, we cannot recover self-shadowing on the caster.*

The user will require a mask of the caster to properly combine the images in compositing. While this helps compositing edits, a serious shortcoming is that the manipulation of self-shadows becomes impossible as they are not picked up.

But the main issue with the image subtraction method is the number of additional renders that must be performed. If there are N shadow casters in the scene, we need to execute a total of $2N + 1$ renders if we count in the original image. In the following, we introduce an approach based on the same intuition that solves both issues.

4.3 • Characterization of shadow in the path space

Our idea to solve the limitations underlined in Section 4.2.3 is to reformulate the image space subtraction method using path integrals.

4.3.1 • Subtraction of the path integrals

Similar to the image space subtraction approach, we assume a single shadow caster of interest $\mathbf{c}$ defined by its surface $\mathcal{M}_\mathbf{c}$. Omitting the dependence on sensor position, the path integral formulation (Section 3.2) expresses a measure I in the original image as

$$I = \int_\Omega f(\bar{x}) \, \mathrm{d}\mu(\bar{x}) \, .$$

The same formulation applies to J the invisible and B the blackbody versions of the scene. In both cases, overriding the BSDF of $\mathbf{c}$ does not change the extent of its surface. Thus, when writing the path integrals that correspond to J and B, the path space Ω and measure $\mathrm{d}\mu$ do not change. The influence of the BSDF is present only in the measurement contribution function f; we denote the two different functions f_J and f_B accordingly:

$$J = \int_\Omega f_J(\bar{x}) \, \mathrm{d}\mu(\bar{x}) \quad \text{and} \quad B = \int_\Omega f_B(\bar{x}) \, \mathrm{d}\mu(\bar{x}) \, .$$

Shadow S is obtained by subtracting J and B:

$$S = J - B = \int_\Omega (f_J(\bar{x}) - f_B(\bar{x})) \, \mathrm{d}\mu(\bar{x}) \, . \tag{4.1}$$

4.3.2 • The set of encountered paths

The integrand of Equation (4.1) is the subtraction of f_J and f_B, but the only difference between these two functions is the BSDF of $\mathbf{c}$. This BSDF ρ is involved precisely when a vertex of light path $\bar{x}$ corresponds to a scattering event on $\mathcal{M}_\mathbf{c}$, the surface of $\mathbf{c}$. For this reason, we introduce $\Omega_\mathbf{c}$ the set of all light paths that encounter $\mathcal{M}_\mathbf{c}$ at least once:

$$\Omega_\mathbf{c} = \{\bar{x} = \mathrm{x}_0\ \mathrm{x}_1 \dots \mathrm{x}_k \mid \exists\, i \in [\![0,\ k]\!], \mathrm{x}_i \in \mathcal{M}_\mathbf{c}\}\ .$$

As $\Omega_\mathbf{c}$ is a subset of Ω, we decompose the integral subtraction in two terms:

$$S = \int_{\Omega_\mathbf{c}} (f_J(\bar{x}) - f_B(\bar{x}))\, \mathrm{d}\mu(\bar{x}) + \int_{\Omega \setminus \Omega_\mathbf{c}} (f_J(\bar{x}) - f_B(\bar{x}))\, \mathrm{d}\mu(\bar{x})\,.$$

In the previous equation, each integral can be simplified:
- Outside of $\Omega_\mathbf{c}$, both f_J and f_B are equal to the original f.
- In the subset $\Omega_\mathbf{c}$, f_B is null as caster $\mathbf{c}$ is a pure absorber.

Applying these observations yields the following path integral:

$$S = \int_{\Omega_\mathbf{c}} f_J(\bar{x})\, \mathrm{d}\mu(\bar{x})\,, \tag{4.2}$$

which reads that shadow is the contribution of all light paths that encounter the caster, considering that the latter is invisible. It is a positive quantity expressing the local loss of radiance over the sensor due to $\mathbf{c}$. This definition is concise and intuitive, and because the integrand is tractable and defined on a subset of Ω, we may now build on Monte Carlo algorithms to estimate the value of shadow layers.

4.4 • Efficient rendering of multiple shadow layers

Our rendering algorithm is based on path tracing (Section 3.2.2), and we describe the changes that enable the measure of shadow at the same time as radiance.

4.4.1 • Algorithm outline

Until now, we have only considered a single shadow caster to carry out our derivations; the algorithm described in this section handles any number of casters $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N$. These N objects are identified from the surface they span in the scene, and are the only required user input. Most renderers allow the association of unique identifiers with surfaces of the scene, and we have added this capability to pbrt-v3 as well (Section 3.3.2). Each shadow caster is associated with a measurement contribution function $f_{J,\mathbf{c}_i}$ and a shadow layer $S_{\mathbf{c}_i}$. The goal of our path tracer is to render image I and $(S_{\mathbf{c}_i})_{i \in [\![1,\ N]\!]}$ in a single pass.

The main difference between a standard path tracer and our implementation is that the contribution of paths is evaluated not only with f, but also each $f_{J,\mathbf{c}_i}$ for i in $[\![1,\ N]\!]$. In Section 3.2.2, we have seen that the sampling of paths in Ω follows a probability density function p that should have the same shape as f to reduce the amount of noise in the image. The existence of multiple measurement contribution functions results in a different sampling scheme that affects two steps of the algorithm: the propagation of paths in the scene, and the sampling of direct light at each vertex.

Path propagation

The propagation of a path starts from the camera, as usual. As long as the path does not encounter one of the shadow casters, the next direction starting from an intersection is chosen according to the BSDF at that location; this corresponds to sampling from f. When direct lighting is gathered, the contribution is recorded in the original image I.

The first time a caster $\mathbf{c}_i$ is encountered, the path being built now belongs to $\Omega_{\mathbf{c}_i}$. The sampling procedure determines which measurement contribution function to use between f and $f_{J,\mathbf{c}_i}$. To perform this choice, we use the *one-sample model* [Vea97]: $f_{J,\mathbf{c}_i}$ is given probability γ, and f probability $1-\gamma$. The constant γ is a parameter of our method, and will be discussed more in depth in Section 4.6.1; by default, we set $\gamma = 1/2$. The remainder of the propagation is conditioned by the outcome.

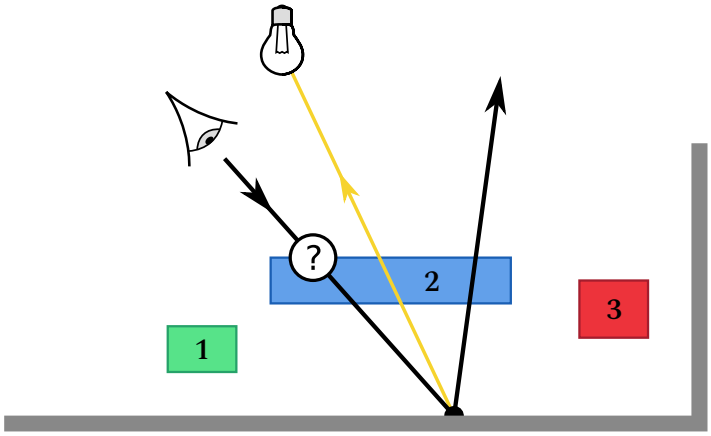
If $f_{J,\mathbf{c}_i}$ is chosen, we say that $\mathbf{c}_i$ becomes the *assigned caster* of the path being built. this has a number of implications on the continuation of the algorithm:
- The image being rendered is the shadow layer: whenever a radiance contribution is picked up, it is accumulated in $S_{\mathbf{c}_i}$.
- The BSDF of the caster is overridden by $\delta(\omega_i+\omega_o)/|\mathrm{n}(x)\cdot\omega_i|$. $\mathbf{c}_i$ thus becomes invisible, and is systematically skipped when encountered again.
- The measurement contribution function cannot change anymore: no further choice is given to the path when intersecting another caster.
- The throughput of the path is divided by γ.

If f is chosen, propagation resumes as usual:
- The image being rendered is still I.
- The BSDF of $\mathbf{c}_i$ does not change, and the path scatters normally on its surface.
- Caster $\mathbf{c}_i$ cannot be considered for assignment anymore: this would involve making it invisible, but a scattering event already exists along the path.
- However, another shadow caster can be considered for assignment later on.
- The throughput of the path is divided by $1-\gamma$.

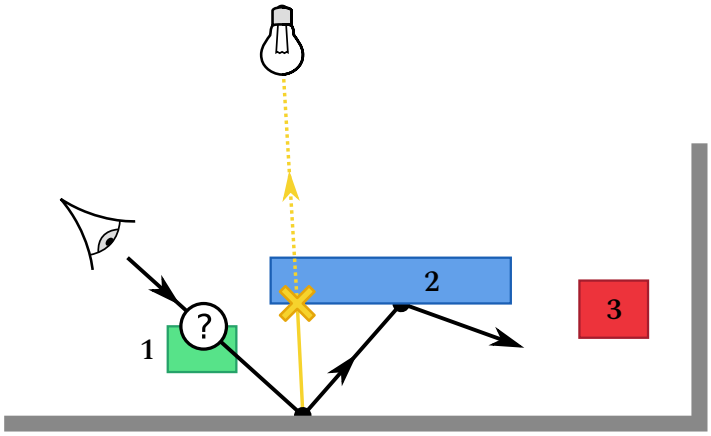
In terms of implementation, the differences with a standard path tracer are minimal. As long as no caster has been assigned, the algorithm only needs to maintain the list of casters that were already encountered. When a caster is assigned, its identifier is simply stored inside a variable, and the path throughput is updated according to γ.

Direct light gathering

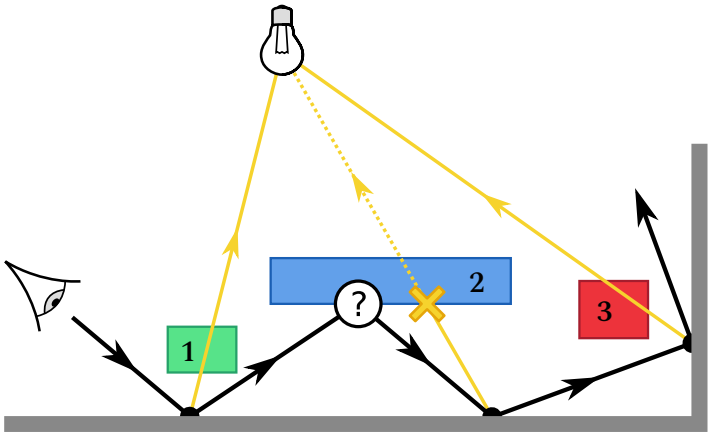
At each vertex, the path tracing algorithm gathers direct lighting from a source picked at random in the scene, and a shadow ray is sent toward the light source to check its direct visibility. In our prototype, this test must remain coherent with the history of the path.
- If the path has an assigned caster $\mathbf{c}_i$, the rules described for propagation apply: $\mathbf{c}_i$ is skipped by the shadow ray, and any radiance contribution goes to $S_{\mathbf{c}_i}$.
- Otherwise, the first occluder in the way is skipped if it is a tagged caster $\mathbf{c}_i$ that has not been discarded for assignment. If the shadow ray eventually connects with the source after a skip, the contribution goes to the corresponding shadow layer $S_{\mathbf{c}_i}$ instead of I.



(a) *The blue caster* ***2*** *is hit and assigned to the path; it is skipped during propagation and shadow ray testing.*



(b) *Caster* ***1*** *is assigned to the path; caster* ***2*** *is thus considered a normal object for both path construction and shadow rays.*



(c) *The ray hits* ***2****, which is not assigned to the path. Further intersection and shadow ray tests consider it a normal object. For shadow rays, other casters are skipped and the radiance contribution goes to their shadow layer.*

Figure 4.7: *Three examples of light path construction.*

A few examples of paths being built by the algorithm are given in Figure 4.7. Many shadow rays that would be occluded during a standard path tracing render are contributing to shadow layers. Thanks to its single pass nature, our solution better leverages the cost of building paths; its performance is quantified more precisely in Section 4.5.2.

However, we see in Figure 4.7b that the green shadow caster is skipped by the primary ray, meaning that the issue underlined in Section 4.2 is still present: shadow is measured over the sensor of the camera, and not on the surface of objects. The resulting shadow layer will display the background leaking through the surface of the green object, and impede editing. In the following, we detail the modifications required to fix this behavior, and show that they bring more artistic control at the same time.

4.4.2 • Enhanced user control

The first benefit of our path tracing algorithm is computational, as it only requires one rendering pass to export all desired layers. In terms of artistic control, we also surpass the subtraction method thanks to several parameters that refine the generated shadow layers.

Measure on catchers

To measure shadows on surfaces of the scene instead of the sensor, we allow the user to specify a list of shadow catchers; if that list is empty, then all objects in the scene are considered catchers. The beginning of propagation behaves exactly like standard path tracing: the path is given no choice to skip casters, and they are not added to the list of encountered objects when intersected; this prevents the measure of shadow on the sensor. The first time a catcher **o** in the list is encountered, skipping casters is allowed starting from the next interaction, and so shadow is measured on the surface of **o**.

If object **o** is also a caster, self-shadows can appear on its surface. Figure 4.8 shows that self-shadowing is non-null on the back of the dragon: this contribution comes from the reflection of the white part on the pedestal. While the ability to render self-shadows is important for editing, they are not always desirable, and we add an option to disable them for a user-provided list of objects. When shadow starts being measured on one of them, it is immediately added to the list of already encountered objects. This way, the path will not be given the choice to skip it when intersected again.

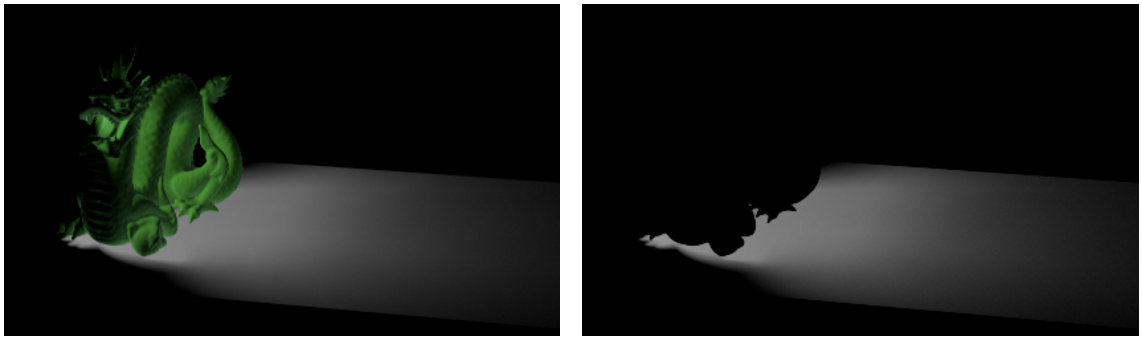


Figure 4.8: *Shadow in the Dragon scene with (left) and without (right) self-shadowing enabled.*



All surfaces are shadow catchers (default behavior) *Mirror removed from the list of shadow catchers*

Figure 4.9: *When a purely specular surface catches shadows, artifacts appear. In these two setups, the left image is the shadow layer and the right one is the result of shadow removal.*

It is also useful to prevent specular surfaces from being be treated as shadow catchers. Otherwise, the measure of shadow on the sensor still takes place, as it is just carried forward to the next bounce. The effect of a mirror on the measure of shadow is illustrated in Figure 4.9: when included in the list of catchers, artifacts appear where the reflection of caster is visible; removing the mirror from the list of catchers solves the issue.

Light groups

Because lighting from different sources is additive, a common operation discussed in Chapter 3 is to separate the contribution of different light sources inside separate layers. By default, shadow layers contain the radiance lost from all light sources in the scene. Because they also have a linear behavior, being able to separate shadow by light source is relevant for artistic editing, to disambiguate overlapping shadows for instance (Figure 4.10). We thus add the option to export multiple shadow layers per caster, and dispatch each light contribution according to user-defined light groups.

Direct and indirect components

Direct shadow location is usually easy to predict from the relative position of the light source and shadow caster. However, indirect shadowing implies multiple light bounces that make it hard to anticipate. For this reason, each shadow layer can further be split into its direct and indirect contributions. The differentiating factor is that indirect shadow is obtained only after a caster has been skipped by the path, and a scattering event happened afterward. In a sense, we say that direct shadow is carried by shadow rays of length $k = 2$, while indirect shadow corresponds to longer shadow rays.

4.5 • Results and performance

The next section presents several compositing examples based on shadow layers obtained using our implementation, and Section 4.5.2 discusses the performance overhead and error convergence compared to a standard integrator.

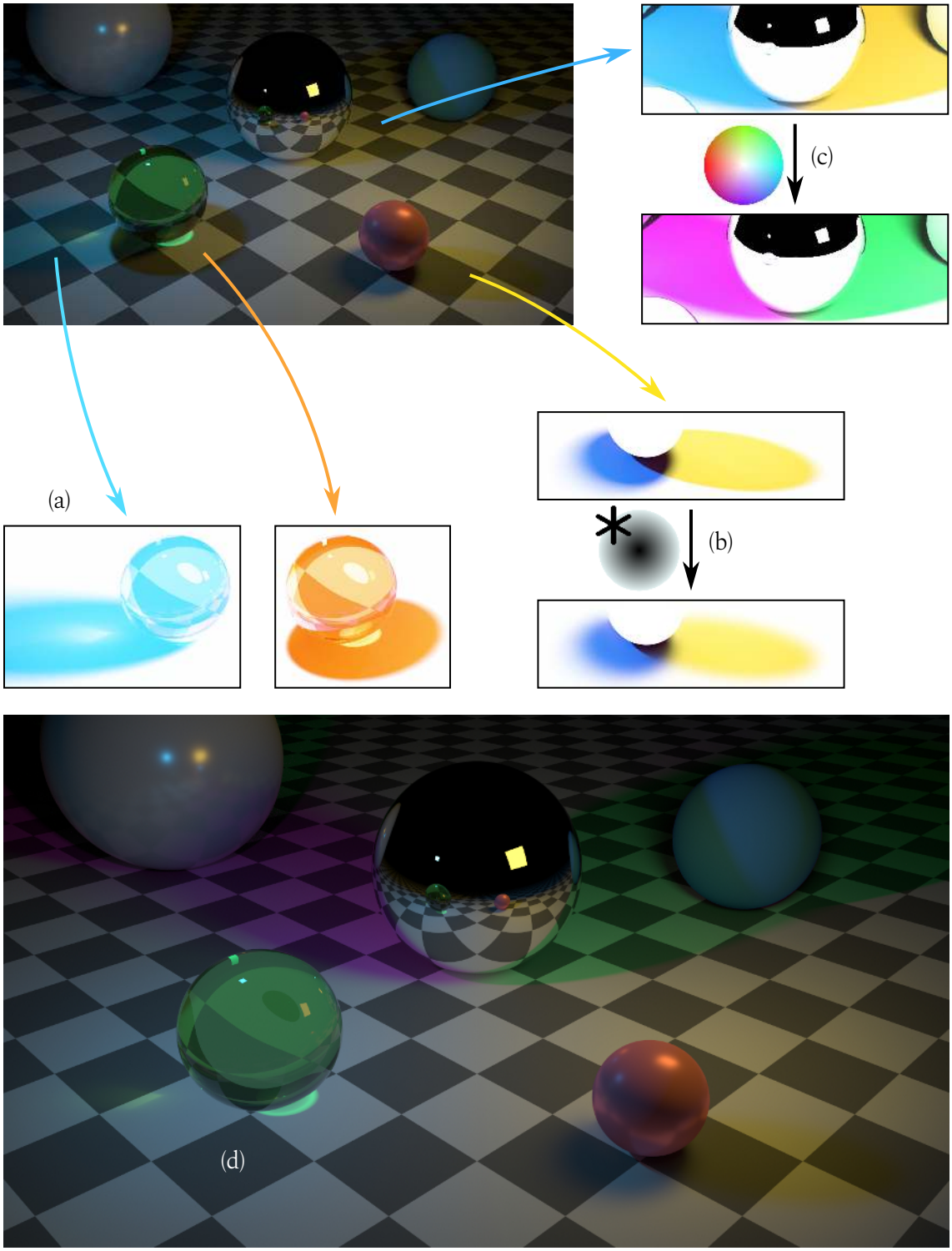


Figure 4.10: *Compositing examples on the Marbles scene; for each edit, we display the shadow ratio defined as $I/(I+S)$. (a) Shadow ratios are separated per light source, and used to carry out common compositing tasks such as (b) shape smoothing, (c) color grading, or (d) removal.*

4.5.1 • Application in compositing

Figure 4.10 shows the *shadow ratio* of each shadow caster in the Marbles scene. The shadow ratio is a useful quantity for artistic editing defined as $I/(I+S)$: it is the quotient between the original image and its shadowless version. The appeal of the shadow ratio is that it takes away the influence of the shadow catcher's diffuse texture. In this scene, the checkerboard pattern is not visible anymore, and we can thus freely apply color filters without changing the aspect of the underlying surface. The shadow ratio is similar to the *visibility ratio* of Obert *et al.* [Obe10], and the V-Ray renderer generates a *raw shadow* AOV following a similar equation for direct shadows [Cha21]. While shadow layers have a linear relationship with the original image, the shadow ratio is a multiplicative quantity.

This setup emphasizes the global illumination context we operate in. It contains two area light sources over the foreground spheres, and shadow layers are separated per light source. After removing the shadow of the green sphere, the result remains coherent despite the caustics. The mirror in the middle also reflects light from the two sources, and creates indirect shadows in the rest of the scene; the hue of its own cast shadow is rotated thanks to the shadow ratio. Finally, self-shadowing is disabled for the red plastic sphere as its shadow is later smoothed by an image space filter.

The Flowers scene of Figure 4.11 confirms that the shadow ratio stays smooth even if shadows are cast on a fine-grained sand texture. It is manipulated using advanced image editing techniques, such as direct painting or a cage deformation to transform the shadow's shape. After the ratio has been changed, it is multiplied with the shadowless version of the image $I+S$ to reintroduce shadows and obtain the composite result.

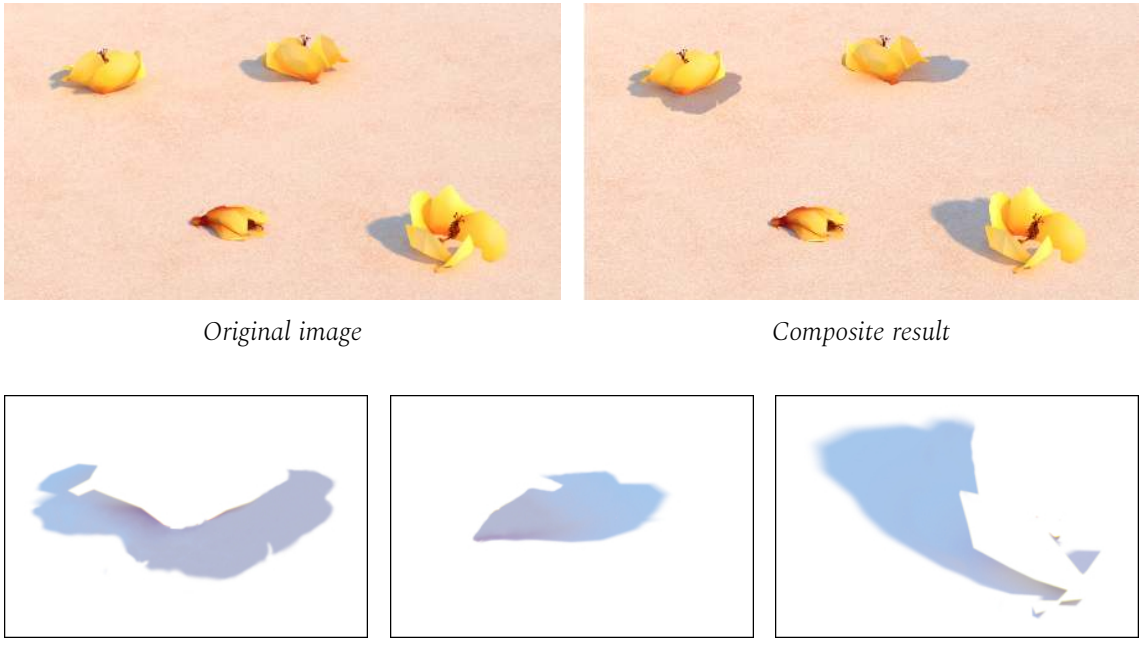
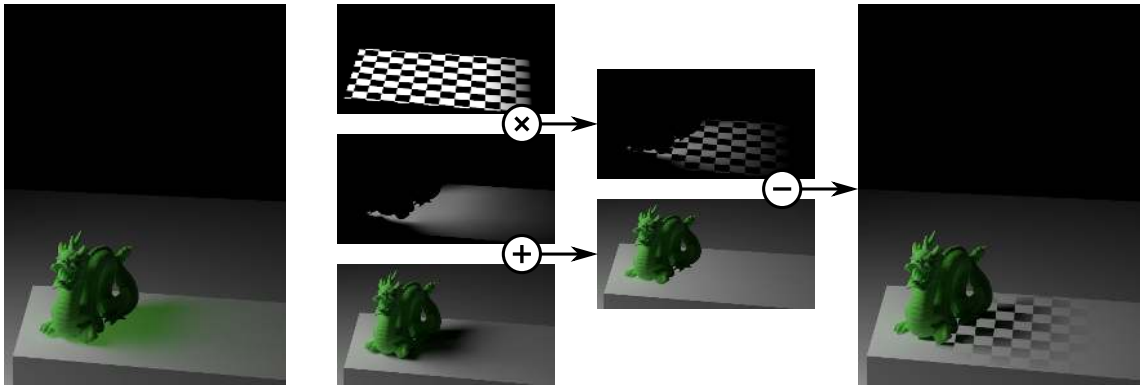


Figure 4.11: *In the Flowers scene, the shadow ratios displayed on the bottom row remain smooth despite the sand texture. They are transformed separately using various image editing techniques.*



Color grading of shadow *Compositing graph corresponding to the inlaying of a checkerboard pattern*

Figure 4.12: *Our approach fully support indirect shadows, as seen here in the Dragon scene.*

Compared to the shadow matte from conventional renderers such as Arnold, our shadow layer correctly accounts for indirect shadows, as seen in Figure 4.12. It is used to color grade the dragon's shadow into a green tint, or inside a more advanced compositing graph that allows the user to inlay a checkerboard pattern over the pedestal.

Shadow layers also help in relighting tasks: the original Moana Island render in Figure 4.13 has an uneven lighting as the sun is occluded by the tall trees. Traditionally, this image would have underwent a histogram correction, a global operation that must be locally fine-tuned afterward. Instead, an addition between the original image and the shadow layers is a practical solution to brighten up the landscape.

4.5.2 • Performance considerations

Our integrator is implemented in pbrt-v3, and we compare it with the original path tracer. The results are presented in Table 4.1. For each scene, N is the number of rendered shadow layers, without per-light source or direct/indirect shadow separation; where $N = 0$, the standard pbrt-v3 path tracer was used. The first column displays the total number of samples at a pixel; this sampling budget is shared among all the rendered layers. The benchmarking machine runs an Intel Xeon E5-2630 v4 processor with 20 threads at 2.20 GHz, and has 64 GiB RAM.

When rendering at least one shadow layer along with the original image, we observe an increase in render time of a factor varying between 1.1 and 1.3. It is mainly due to the additional intersection tests that must be performed when skipping a caster repeatedly during propagation or direct light gathering. As N increases, we observe that the render time keeps increasing slightly. Indeed, managing several images takes some additional time, for example when looping over the reconstruction filter's support after each measure. Storing additional layers has a predictable footprint on memory, costing the equivalent of a full resolution image per layer.

However, a single path contributes to different layers depending on its propagation. This variation in the number of radiance contributions that each image receives means that time alone is not a sufficient factor for comparison. Assessing render quality using

Root-Mean-Square Error (RMSE) or relative RMSE is pessimistic on the shadow layers, as they mostly contain null values. We thus compute the Structural Similarity (SSIM) [Wan04] relative to a reference render involving at least 16 times more samples, with a radius of 5 and $\sigma = 1.5$. A SSIM of 1 means indistinguishable images, and 0 no similarity. Given a fixed sample budget, SSIM decreases with the number of shadow layers being rendered. When many shadow layers span a large image space support, all images particularly suffer from under-sampling.

We also provide the percentage of Zero Radiance Paths (ZRP) that are constructed but eventually discarded as they bring no energy. This percentage systematically decreases with increasing N, which confirms that computing all images in one pass allows us to better leverage the cost of building paths.

While fixing the sampling budget and activating the rendering of shadow layers corresponds to a typical use case, we study more in-depth the interaction of the three variables involved: sample count, rendering time and pixel variance. To do so, we fix time or variance in the original image to a target and compare our implementation to standard path tracing; results are found in Table 4.2. For fixed variance, pixels can adaptively use up to 4 times the initial sampling budget of Table 4.1.

In the Marbles scene (Figure 4.10), fixed-time performance is comparable to that of Table 4.1. However with a target variance, both integrators consume additional samples in regions with difficult light transport such as the caustics under the glass sphere, where our algorithm also picks up indirect shadows. The Dragon scene (Figure 4.12) only contains indirect lighting and shadows that cause very slow convergence for both algorithms, as they rely on path tracing. Whereas the Flowers view from Figure 4.11 contains mostly direct shadows that have little performance impact, the full Moana Island scene (Figure 4.13) is the most challenging: across the deep foliage, many intersection tests are performed to recover missing light from the sources on the other side.

4.6 • Advanced implementation guidelines

We have presented a path tracing implementation that covers most use cases and demonstrated its capabilities on a production-grade scene. In the following, we show how to fine-tune its behavior using additional parameters, and provide an overview of possible implementations for popular integrators other than path tracing.

4.6.1 • Performance parameters

The assessment of our prototype's performance has underlined two sources of degradation. First, estimation error increases with the number of rendered layers as the contribution of light paths are now dispatched among them. Second, performing additional intersection tests when casters are skipped incurs a measurable overhead in rendering time. We propose two performance parameters for better control over the rendering process.



Figure 4.13: *Top: in the original view of the Moana Island scene, the Hibiscus (yellow flowers) and Pandanus (elongated leaves) are shadowing other plants. Middle: the sum of their shadow layers contains the radiance lost on the rest of the scene. Bottom: thanks to a single image space addition, lighting in the image is balanced with little user intervention.*

Scene	N	Samples	Time	SSIM	ZRP
Marbles (Figure 4.10)	0	2048	15' 27"	0.903	26%
	1	2048	17' 27"	0.901 / 0.983	17%
	2	2048	18' 07"	0.896 / 0.956	12%
	3	2048	18' 28"	0.896 / 0.956	10%
Flowers (Figure 4.11)	0	256	03' 19"	0.901	15%
	1	256	03' 28"	0.899 / 0.838	12%
Dragon (Figure 4.12)	0	4096	27' 53"	0.927	91%
	1	4096	33' 33"	0.889 / 0.952	90%
Moana Island (Figure 4.13)	0	1024	34' 30"	0.992	87%
	1	1024	42' 35"	0.992 / 0.996	81%
	3	1024	46' 05"	0.992 / 0.875	80%
	5	1024	47' 20"	0.992 / 0.864	79%

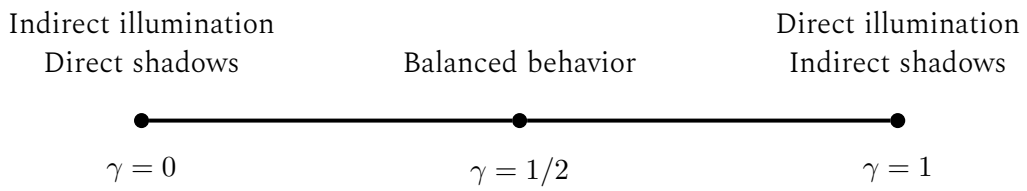
Table 4.1: *Performance results for the render of N shadow layers. The overhead between $N = 0$ and $N = 1$ is mainly due to additional intersection tests. As N increases, the various images are less converged compared to the reference, and their handling takes processing power. The SSIM is given for the original image / the shadow layer with minimum similarity to the reference.*

Scene	N	Time	SPP	$\overline{\text{Var}}$	Var	Time	$\overline{\text{SPP}}$
Marbles (Figure 4.10)	0	15'	2048	0.003	0.01	18' 24"	2321
	1	15'	1824	0.004	0.01	20' 08"	2318
	2	15'	1728	0.004	0.01	21' 46"	2360
	3	15'	1696	0.004	0.01	22' 14"	2371
Flowers (Figure 4.11)	0	10'	832	0.271	0.5	4' 01"	291
	1	10'	784	0.271	0.5	4' 15"	291
Dragon (Figure 4.12)	0	30'	9856	0.153	0.1	34' 58"	8122
	1	30'	8256	0.153	0.1	41' 44"	8139
Moana Island (Figure 4.13)	0	30'	960	0.007	0.01	39' 36"	1277
	1	30'	768	0.007	0.01	51' 11"	1289
	3	30'	704	0.008	0.01	54' 16"	1310
	5	30'	704	0.008	0.01	55' 56"	1317

Table 4.2: *The middle column shows the number of samples contributing to all layers, and the mean variance of the original image for equal-time runs. The right column shows the computation time for a target pixel variance in the original image, and the average sampling.*

Skip probability

An important parameter of the method is γ, the probability of a caster being skipped the first time it is encountered during propagation. We have seen that skipping casters when building paths is the key to rendering indirect shadows. Yet, scattering on objects is also required to properly render indirect illumination. This means that γ controls the balance between indirect shadows and indirect illumination:



For an unbiased rendering of the original image and all shadow layers, it is mandatory that $0 < \gamma < 1$. In this range of values, γ will exclusively affect variance and thus noise in the rendered images.

In some scenes, skipping casters during propagation is not required: a convex subject casting shadow on a plane under direct lighting such as the Simple Sphere (Figure 4.1a) will never create indirect shadows. In this kind of setup, the algorithm spends time creating paths that skip a caster seeking for indirect shadow from a light source, but do not bring any contribution. While it requires a good understanding of shadow formation, we make γ available as a parameter to fine-tune rendering.

Maximum number of skips

When a caster is assigned to a path, it is considered as invisible and systematically skipped when encountered; this results in additional intersection tests, especially in the presence of fine geometry. An example is the Pandanus tree in Figure 4.13, where the mesh model contains thousands of hanging leaves. To obtain the shadow of this tree, our prototype intersects leaf after leaf, searching for a light source or an intersection with another object. Parallelism is compromised as this region exhibits more complex geometry than the rest of the image, and overall performance is seriously affected.

The same problem arises in standard path tracing if the depth of paths is not controlled; in this case, two well-known techniques remedy the problem:

- A hard threshold on the depth is applied in the rendering loop, which introduces bias in the image. The bias decreases as the threshold increases.
- Russian roulette [Vea97] is used to terminate paths randomly without introducing bias. For the roulette to be efficient, the termination probability is proportional to the inverse of a path's throughput so that paths with low contribution are cut early on.

The main difference in our situation is that contrary to scattering, skipping an invisible caster never changes the throughput of a path, meaning that Russian roulette does not generalize. We thus need to set a threshold on the maximum number of skips along a path. This may cause bias in the shadow layers, similar to that of the original image when a depth threshold is applied. By default, we set the maximum number of skips to twice



Original image *Shadow layer, 6 skips* *Shadow layer, 20 skips*

Figure 4.14: *If the maximum number of skips is too low, the shadow layer displays characteristic bias where paths are terminated before reaching a light source.*

the maximum depth. The rationale is that two skips are required to enter and exit an enclosing surface, and we count one such encounter per scattering interaction with an object. The threshold is available as a parameter so that the maximum number of skips can be increased if shadow layers appear biased, as in Figure 4.14.

4.6.2 • Shadow layers with other integrators

While the subtraction method (Section 4.2) works with any rendering algorithm, our implementation only applies to path tracing. Nevertheless, a general rule is that the algorithm must be able to sample paths and compute their contribution according to a new set of measurement contribution functions (Section 4.3). This has a number of implications according to the chosen integration scheme.

Bidirectional path tracing

With Bidirectional Path Tracing (BDPT, [Vea97]), two subpaths are sampled successively: a camera subpath starts from the sensor, similar to path tracing, and a light subpath starts from a light source. Two random walks in the scene are performed to build them.

Assuming that the camera subpath is sampled first, it must follow the same behavior as described in Section 3.3.2: a shadow caster can be assigned and skipped during propagation. The direct light gathering aspect is not relevant here, as it is assumed by the light subpath. If a caster was assigned to the camera subpath, the light subpath inherits it; otherwise, assignment is still open for all casters that were never encountered.

The specificity of BDPT is that all possible pairs of subpaths prefixes are connected to form full paths. During this step, the propagation history of each subpath prefix determines the contribution of the full path, and were it goes:

- If both prefixes have never skipped a caster, they are connected and weighted as usual and the resulting contribution goes to the original image I.
- If at least one of the prefixes has skipped a caster, the connection ignores it when testing visibility between the two prefix endpoints. The radiance contribution is divided by γ, and goes to the shadow layer of the skipped caster.

Photon mapping

The rendering of shadow layers also generalizes to photon mapping and its later iterations [Hac09]. Our implementation recommendations mostly follow those of path tracing, adapted to the photon pass. The generation of hit points during the eye pass does not change: casters are not skipped when starting from the camera, as we prevent the measure of shadow on the sensor and mirror surfaces (Section 4.4.2).

Metropolis light transport

In the Metropolis algorithm, rendering begins with a set of warm-up paths usually obtained with BDPT. Afterward, these paths undergo either small mutations to locally explore the path space Ω (shifting a vertex position), or large mutations to ensure that all parts of Ω are reached (changing the path length). The acceptance of these mutations depends on the relative change in throughput they incur, which selects mostly advantageous mutations and explains why the distribution of samples is proportional to the measurement contribution function f for an infinite number of mutations [Vea97].

First, the set of warm-up paths must be obtained with a technique adapted to shadow layers, such as the version of BDPT presented earlier. Second, while existing mutations still apply, their acceptance probability must now take into account the different measurement contribution functions: f, but also each $f_{J,\mathbf{c}_i}$. This specificity is costly to take into account during acceptance, and prevents the convergence of the samples distribution because there are now multiple target probability density functions.

For this reason, we reckon that a straightforward solution is to consider the chosen measurement contribution function as a characteristic of the path, and to add a large mutation that corresponds to a random change in the chosen function. This way, successive correlated samples obtained via small mutations have the most chance of being accepted so that the local exploration of Ω remains possible and efficient.

4.6.3 • Shadow path expressions in Arnold

The variations of popular Monte Carlo integration schemes that we have described above sample paths that carry shadow in addition to those carrying radiance. This enables the use of light path expressions (or rather, *shadow path expressions*) to export shadow layers.

Seeking to close the gap between our prototype and the end users of our method, we have implemented shadow layers based on LPEs in Arnold 6.2 for Maya. Since version 5.0, the appearance of objects in Arnold is ruled by *closures*, functions that describe the scattering of light at every interaction [Aut21a]. Arnold being a path tracer, we have applied the changes described in Section 4.4, adapted to the closures system:

- Path propagation is altered by adding a straight transmission to the original list of closures on the surface of shadow casters. If Arnold samples it, the straight transmission closure will match the event scattering type **s** in LPEs.
- Every time direct lighting is gathered, we add fictitious emission closures **E** to recover the amount of radiance lost due to shadow casters.

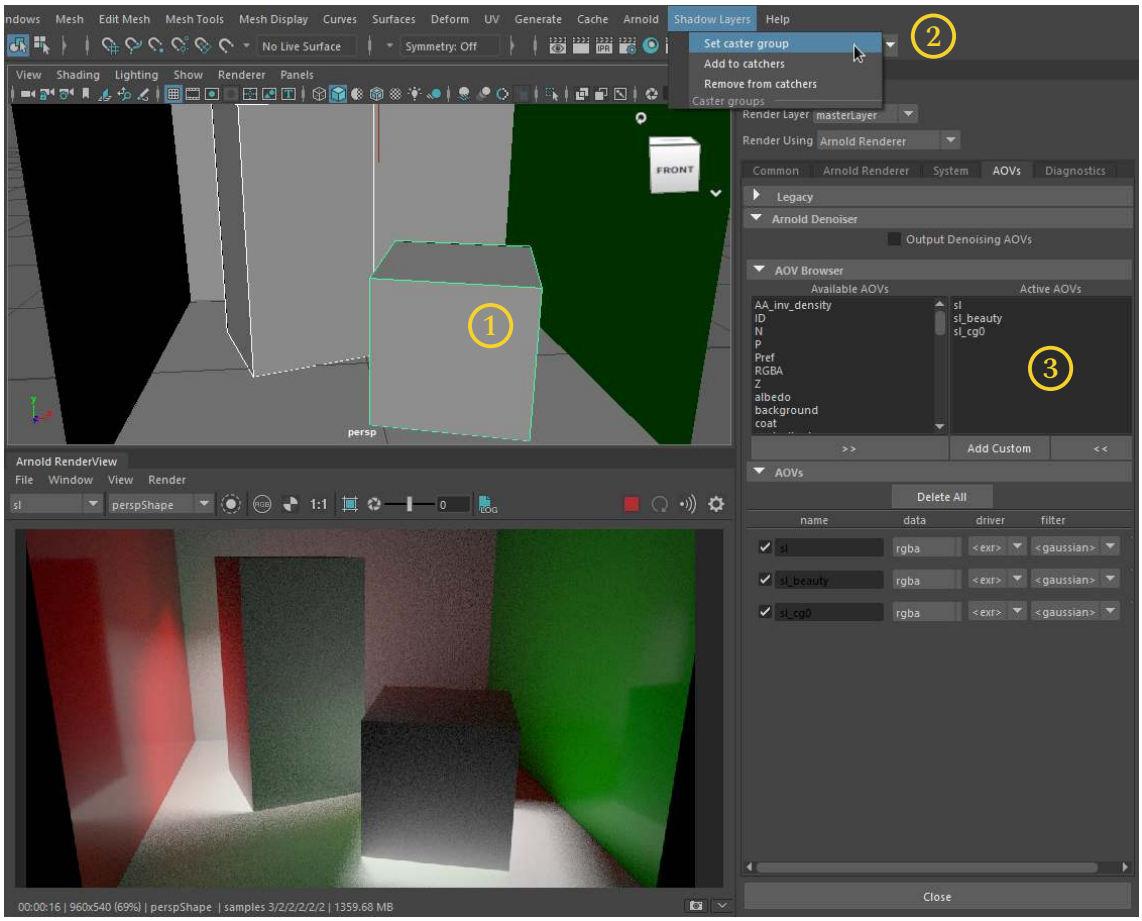


Figure 4.15: *Our Arnold for Maya implementation is based on light path expressions, and works hand in hand with the closures system that was added in version 5.0 of the renderer [Aut21a]. ① The user starts by selecting several objects and ② assigning them to a shadow caster group. ③ The plugin automatically sets up the corresponding LPEs inside AOVs: a new beauty layer with correct shadows, a shadow layer for all casters groups, and a shadow layer per caster group.*

The new events are tagged using object identifiers to separate the shadow of different casters. An unsettling change with this choice of implementation is that the beauty layer corresponding to all light paths `.*` is now entirely without shadows; the original render must be requested using a light path expression that ignores our straight transmission and fictitious emission events. An overview of our Maya plugin is shown in Figure 4.15.

4.7 • Limitations

In addition to the performance overhead discussed in Section 4.5.2, the method presented in this chapter has several practical limitations that we underline below.

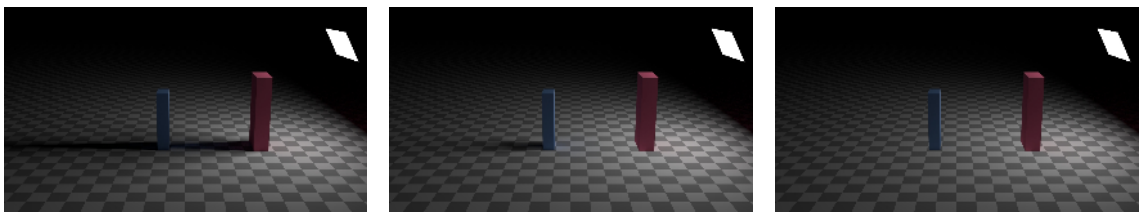
Invisible objects

The rendering of shadow layers is based on replacing the caster's BSDF with a straight transmission $\delta(\omega_i + \omega_o)/|\mathrm{n}(x) \cdot \omega_i|$. If the surface of the shadow caster is already materialized by such a Dirac distribution, it will not create shadow as incoming radiance passes through. Yet, our formulation does measure non-null shadow for these objects.

While straight transmitting surfaces are not physically realizable, they are often used in 3D scenes to hide area lights, as the boundaries of volumetric objects, or to represent portals that emit light or influence its propagation [Sub17]. Realistically, invisible objects will not be tagged as shadow casters by the user. However, if need be, a practical workaround is to use *BSDF flags* to prevent straight transmissions from being considered by the algorithm. These flags are found in all Monte Carlo-based renderer as they are necessary to properly handle the integration of Dirac distributions.

Shadow coupling

Because light paths have a single assigned caster at a time, shadows created by more than one object create inconsistencies, as shown in Figure 4.16. The part of shadow resulting from two occlusions cannot be assigned to only one of the casters, and thus cannot be properly removed during compositing. To circumvent this issue, a solution is to export a shadow layer where both casters are considered as a single object. However, tracking interactions between any two casters among N yields $\binom{N}{2} = N(N-1)/2$ additional layers. Likewise, accounting for all possible interactions involves a total of 2^N layers, resulting in an exponential complexity.



Original image *Adding both shadow layers* *With the two casters as one object*

Figure 4.16: *Our method accounts for shadows created by a single object. In the presence of multiple occlusions, inconsistencies appear when trying to remove shadow in the original image.*

Participating media

To obtain shadow layers for participating media, we could follow the same reasoning as for solid objects and begin with the subtraction method. However, shadow layers obtained by subtraction must be paired with a mask of the caster for proper compositing. With volumetric objects, the concept of mask does not generalize easily as a medium interacts with primary rays at various depths. A solution is to resort to *deep images*: instead of storing a single color value at a pixel, deep images contain a collection of slabs defined by a front depth, back depth, color, and opacity. This additional information is sufficient to correctly compose images with translucent objects or volumes.

Still, the overhead of the subtraction method is the main reason that motivates our path integral formulation. Generalizing it to participating media is not straightforward, as the equations ruling light transport in the presence of volumes are much more involved than the rendering equation. Notably, they involve a new factor known as *transmittance* that represents the amount of radiance lost along rays traversing participating media.

4.8 • Conclusion

Motivated by the need for a solution to edit shadows in compositing, we have started this chapter by investigating the capabilities of existing renderers. We have seen that their option to turn off shadows overestimated lighting, and that a shadow matte has no physical signification and fails to capture shadows created by indirect light sources.

After having defined the different entities involved in the apparition of shadows, we have explored the phenomenon based on our intuition of negative impact. This lead us to a straightforward method to obtain shadow layers, that contain the radiance lost due to an object on the surface of the sensor, based on an image space subtraction between two specific renders of the scene. However, this initial approach has strong limitations in terms of usability and performance.

By translating the subtraction to the path integral formulation, we were able to implement a path tracing algorithm that renders the original image and any number of shadow layers in a single pass, with enhanced user control on the exported layers. We demonstrated two prototypes based on pbrt-v3 and Arnold for Maya, and their use in compositing throughout several example scenes.

Two questions remain open. First, we considered shadow casters in isolation from one another: when light is occluded by several casters, the corresponding shadow is not picked up. Second, no generalization from solid objects to participating media seemed satisfactory. The the next chapter presents a solution to alleviate these two shortcomings.

Chapter 5

Generalized Shadow Layers

So far, we have described the measure of shadow for solid objects, and without considering their mutual interactions. However, participating media do cast shadows as they interact with light rays, and multiple occlusions from different objects along a light path create a mutual shadow that is not taken into account using the current approach.

Recovering and editing the shadow of volumetric objects saves precious time in production, where physical fluid simulations are now common. Participating media are the correct models to represent liquids and are staples of special effects departments to depict explosions, smoke, clouds, or fog. For example, when a simulated explosion is inserted inside a real footage, its cast shadow must closely match the rest of the scenery for a convincing result. The merging of the two sequences is typically overseen by lighting artists and finalized in compositing, where shadows are reworked to achieve a plausible look.

The need for a generalized formulation of shadow layers is the motivation of this chapter, where we address the extension of shadow layers to participating media and multiple occlusions along light paths (Figure 5.1).

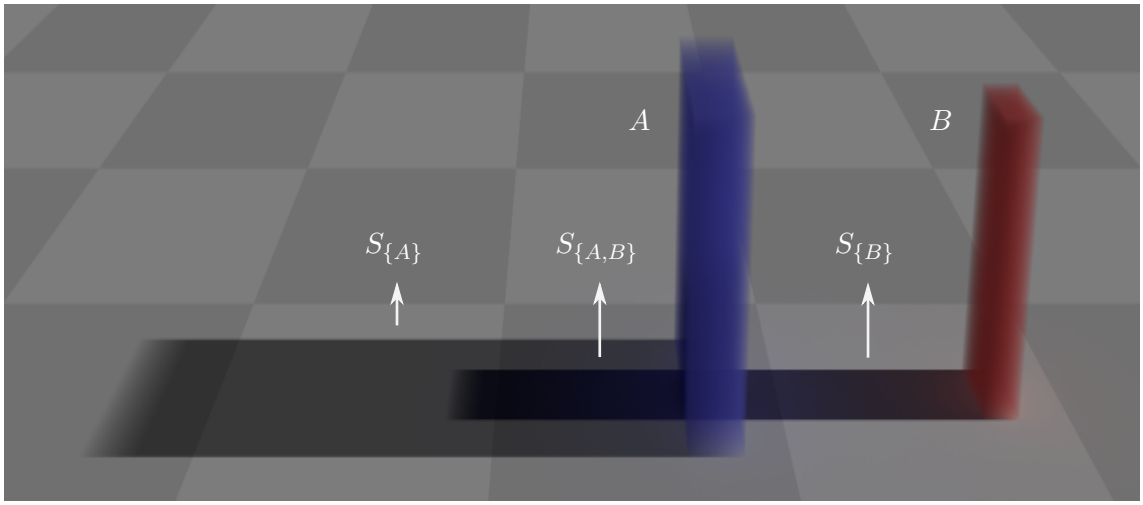


Figure 5.1: *For now, shadow layers are associated with a single solid object. In this chapter, we generalize the method to participating media and mutual shadows such as* $S_{\{A,B\}}$.

To approach volumetric light transport, we start by analyzing the negative impact of a single participating medium of interest in Section 5.1. Negative impact is related to the amount of radiance lost along rays that traverse the medium, and we quantify it based on the physical equations that rule volumetric light transport. However, this characterization of impact is local, and not does account for multiple occlusions along a path.

For a more robust solution, Section 5.2 introduces a version of the path integral formulation that supports participating media. We use it to translate the quantification of impact from light rays to paths, and to consider multiple occlusions by different casters. This leads us to a path integral formulation that measures the shadow of any number of solid or volumetric casters. Section 5.3 proves that this new integral is a proper generalization of Equation (4.2), as the two coincide for a single solid shadow caster.

This generalized formulation serves as the basis to implement an integrator based on path tracing in Section 5.4, from the modification of two key steps: path construction and direct light gathering. It renders the original image and all shadow layers in a single pass, and exposes a number of parameters to artistically control layers and fine-tune rendering performance. We show several results from our prototype implementation and assess the corresponding rendering times and estimation error in Section 5.5.

5.1 • Radiance loss along a ray

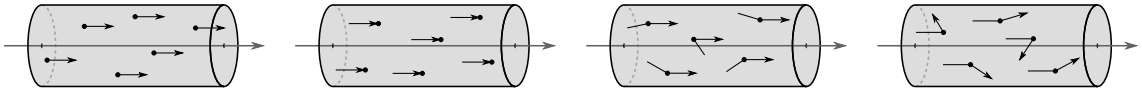
Starting from now, the scenes we consider may contain participating media in addition to solid objects. In this section, we focus on a single participating medium of interest **c** placed in the scene, and analyze how it affects the radiance carried by light rays. Geometrically, a ray starting from point x in space and propagating along ω is described by all points R obeying the parameterization

$$R(r) = x + r\omega\,,$$

with $r \geq 0$ the distance between x and $R(r)$; we omit the dependency of R on x and ω. When expressing radiance at points of the ray, we denote it $L_R(r) = L(x + r\omega, \omega)$. The variation of L around x in direction ω is given by the directional derivative of L_R:

$$\omega \cdot \nabla L(x,\omega) = \frac{L(x + \mathrm{d}r\,\omega, \omega) - L(x,\omega)}{\mathrm{d}r} = \left.\frac{\mathrm{d}L_R}{\mathrm{d}r}\right|_{r=0}. \tag{5.1}$$

When the point x is located within a participating medium, there are several reasons why L changes around x, which are illustrated in Figure 5.2.



(a) *Emission* (b) *Absorption* (c) *In-scattering* (d) *Out-scattering*

Figure 5.2: *Four different types of volumetric events change radiance along the direction of propagation. Energy gains correspond to (a) and (c), while losses are caused by (b) and (d).*

On one hand, energy is gained when the volume emits light spontaneously, or if photons coming from another direction are in-scattered by medium particles toward ω. These contributions are summarized in a source term $Q(x, \omega)$, that corresponds to the positive part of the medium's impact on the radiance field. On the other hand, radiance is lost when some of the photons are absorbed by the medium, or if photons travelling along ω are out-scattered toward an outside direction. The amount of absorbed energy is proportional to the absorption coefficient $\sigma_a(x)$, and the out-scattered radiance to the scattering coefficient $\sigma_s(x)$. These two coefficients are characteristic of the medium, and their sum is called the attenuation coefficient: $\sigma(x) = \sigma_a(x) + \sigma_s(x)$; it governs the negative impact of the medium. The directional derivative of radiance in Equation (5.1) is fully determined by the gain and loss terms, leading to the radiative transfer equation [Cha60]:

$$\omega \cdot \nabla L(x, \omega) = Q(x, \omega) - \sigma(x) L(x, \omega) \,. \tag{5.2}$$

Equation (5.2) is a local description of impact, the radiance changes induced by an infinitesimal slab of participating medium. It is also a differential equation of unknown L that can be integrated between two points. We introduce $z = t(x, -\omega)$ the point located on the nearest visible surface from x in direction $-\omega$, as seen in Figure 5.3. The result of the integration between x and z is the volume rendering equation [Arv93]:

$$L(x, \omega) = \int_x^z T(x, y) Q(y, \omega) \, \mathrm{d}y + T(x, z) L^s(z, \omega) \,. \tag{5.3}$$

Factor T is the transmittance, which quantifies the absorption and out-scattering of radiance between two points. In exponential media, it has the form:

$$T(x, z) = \exp\left(- \int_x^z \sigma(y) \, \mathrm{d}y \right) . \tag{5.4}$$

The boundary term L^s of Equation (5.3) represents the radiance leaving point z in direction ω. It is comprised of surface emission and incoming light that is reflected or transmitted according to the BSDF ρ, integrated over $\mathcal{S}^2$ the sphere of directions:

$$L^s(z, \omega) = L_e(z, \omega) + \int_{\mathcal{S}^2} \rho(z, \omega_i, \omega) \, |\mathrm{n}(z) \cdot \omega_i| \, L_i(z, \omega_i) \, \mathrm{d}\omega_i \,.$$

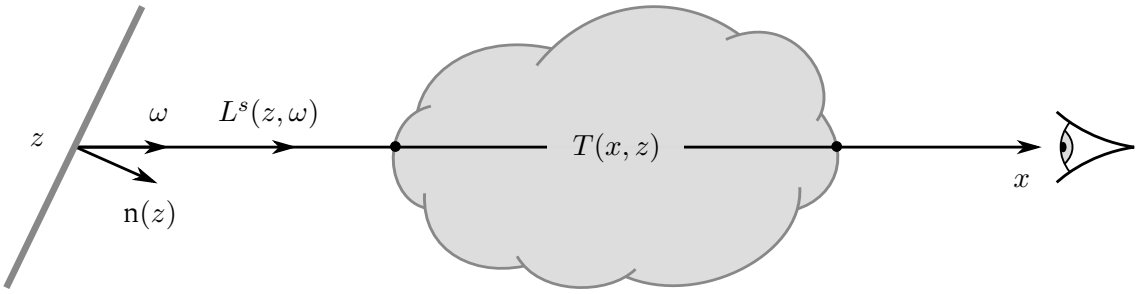


Figure 5.3: *The volume rendering equation is the integrated version of the radiative transfer equation. It describes the exchanges of light along the ray between x and surface point z.*

To quantify the amount of energy lost along the ray when traversing the medium, we follow the intuition of the subtraction method (Section 4.2) and consider the light field when the shadow caster is removed from the scene, *i.e.* no volumetric interaction happens between **c** and the ray. In this case, radiance is given by the rendering equation (3.1) and expressed concisely using the previous notation:

$$L(x, \omega) = L^s(z, \omega)\,. \tag{5.5}$$

The energy missing due to medium interactions is found by comparing Equation (5.3) and Equation (5.5). In Equation (5.3), the integral along the ray is a purely positive contribution from the source term Q that sums up the positive impact of the medium. It is not involved in the formation of shadows, that correspond to the negative component of impact. The energy loss is located in the second term: only a fraction $T(x, z)$ of the radiance outgoing from the surface at z reaches x after absorption and out-scattering have been taken into account. In comparison to the rendering equation, we thus conclude that negative impact along the ray is proportional to the complementary of transmittance $1 - T(x, z)$.

However, our characterization of impact is once again local as it applies along rays. Generalizing it to an entire light path is non-obvious as $1 - T$ is not multiplicative: after two medium traversals, $1 - T(x, z) \neq (1 - T(x, y))(1 - T(y, z))$, meaning that $1 - T$ does not expand naturally into products. Although the previous analysis would be sufficient for a single convex participating medium under direct lighting, the measure of shadow in the general case is based on the path integral formulation of light transport.

5.2 • Generalized path integral formulation

Based on the previous analysis of negative impact from a single participating medium, two generalizations are required to reach an applicable solution:

- Translate our observations from rays to paths despite the non-multiplicativity of $1 - T$.
- Allow the scene to contain multiple participating media, and account for shadows created by several occlusions from different objects.

As was the case in Chapters 3 and 4, the path integral framework is the key to carry out our derivations. We start by presenting its original formulation in the presence of participating media, and perform several modifications to enable the estimation of shadow.

5.2.1 • Classic expansion with participating media

The path integral formulation adapted to scenes that contain participating media has the same form as the previous, and expresses a radiance measure I at the sensor position j as an integral over Ω the set of all light paths: $I_j = \int_\Omega f_j \, \mathrm{d}\mu$ (Figure 5.4). The main changes from Veach's seminal work [Vea97] are located in the measurement contribution function f and measure μ, as devised by Pauly *et al.* [Pau00]. The differential measure $\mathrm{d}\mu$ of a light path $\bar{x} = \mathrm{x}_0 \ldots \mathrm{x}_k$ now develops into the product of the usual surface or volumetric measures, depending on where each vertex x_i is located:

$$\mathrm{d}\mu(\mathrm{x}_i) = \begin{cases} \mathrm{d}A(\mathrm{x}_i) & \text{if } \mathrm{x}_i \text{ is on a surface} \\ \mathrm{d}V(\mathrm{x}_i) & \text{if } \mathrm{x}_i \text{ is in a medium}\,. \end{cases}$$



Original image *Shadow layer of the smoke* *Shadow of the smoke removed*

Figure 5.4: *The path integral formulation extends to scenes that contain participating media. As shown on the right, it allows us to render shadow layers for volumetric objects.*

The measurement contribution function f_j is now defined as:

$$\begin{aligned} f_j(\bar{x}) = {} & L_e(\mathrm{x}_0, \mathrm{x}_1) W_e^{(j)}(\mathrm{x}_{k-1}, \mathrm{x}_k) G(\mathrm{x}_0, \mathrm{x}_1) V(\mathrm{x}_0, \mathrm{x}_1) T(\mathrm{x}_0, \mathrm{x}_1) \\ & \cdot \prod_{i=1}^{k-1} F(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1}) G(\mathrm{x}_i, \mathrm{x}_{i+1}) V(\mathrm{x}_i, \mathrm{x}_{i+1}) T(\mathrm{x}_i, \mathrm{x}_{i+1}) \,, \end{aligned} \tag{5.6}$$

where $L_e(\mathrm{x}_0, \mathrm{x}_1)$ is the radiance emitted from x_0 toward x_1, and $W_e^{(j)}(\mathrm{x}_{k-1}, \mathrm{x}_k)$ is the importance leaving the sensor from x_k toward x_{k-1}. The transmittance T that was defined in Equation (5.4) is multiplied with the visibility term V, which equals 1 if the two points are mutually visible, and 0 otherwise. The other terms depend on the vertices location:

$$G(x, y) = \frac{D(x, y) D(y, x)}{\|x - y\|^2} \quad \text{with}$$

$$D(x, y) = \begin{cases} \left| \mathrm{n}(x) \cdot \frac{x - y}{\|x - y\|} \right| & \text{if } x \text{ is on a surface} \\ 1 & \text{if } x \text{ is in a medium} \,, \end{cases}$$

and $F(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1})$ corresponds to either the BSDF on a surface, or the phase function inside a medium. The phase function is characteristic of a medium, and is similar to the BSDF for surfaces in that it relates incoming and outgoing radiance at each point.

5.2.2 • Generalization to the measure of shadow

The main advantage of the path integral formulation is that all interactions between a light path and objects of the scene, solid or volumetric, are handled at once by the measurement contribution function. This will allow us to overcome the non-multiplicativity of $1 - T$, and to account for several occlusions along light paths. In order to handle multiple shadow casters, we consider $\mathbf{O}$ the set of all objects in the scene. We assume that any interaction along the path is associated with an element of $\mathbf{O}$, and that the objects in the scene are in finite number. Among them, we wish to measure the shadow layer of a subset $\mathbf{C} \subset \mathbf{O}$. As illustrated in Figure 5.1, accounting for sets of shadows casters is necessary to pick up

radiance lost from occlusions by different objects. Starting from the classic formulation presented above, we need to generalize the observations presented in Section 5.1 to express the shadow of $\mathbf{C}$ as a path integral.

Domain of integration

To quantify the radiance loss along a ray, we did not consider the integral term of Equation (5.3) as it corresponds to positive impact from the source term Q. In the path integral framework, this gain is measured by light paths containing at least one volumetric interaction with the medium. More generally, the set of light paths encountering an object $\mathbf{o}$ measures the radiance gains brought by $\mathbf{o}$ to the sensor (Chapter 3). In order to measure shadow, we need to ignore radiance gains and the light paths carrying them. We thus change the domain of integration compared to Equation (5.8): we define $\Omega_{\mathbf{C}}$ the set of all paths with at least one interaction on a caster in $\mathbf{C}$, and integrate over $\Omega \setminus \Omega_{\mathbf{C}}$. In short, we only consider light paths that never encounter an object in set $\mathbf{C}$.

Complementary of the loss factors

Between two successive vertices of a light path, media may decrease the transmittance T while solid objects may cancel the visibility term V. In Equation (5.6), the product VT handles both cases; we call it the *loss factor*. The analysis of Section 5.1 has shown that the fraction of radiance lost when traversing a medium amounts to $1-T$. The same observation applies to factor VT, and indicates that the fraction of radiance lost in the presence of a solid or volumetric object is $1 - VT$. As for transmittance, this complementary is not a multiplicative quantity and we cannot directly accumulate products of $1 - VT$ over an entire light path. This is where we leverage the path integral formulation to group all the loss factors, and only then take the complementary of their product.

Knowing that objects in $\mathbf{O}$ are uniquely identified, we begin by isolating the radiance loss due to each object $\mathbf{o} \in \mathbf{O}$ in Equation (5.6) using the subscript notation $V_{\mathbf{o}}T_{\mathbf{o}}$:

$$\prod_{i=0}^{k-1} V(\mathrm{x}_i, \mathrm{x}_{i+1})T(\mathrm{x}_i, \mathrm{x}_{i+1}) = \prod_{\mathbf{o}\in\mathbf{O}} \prod_{i=0}^{k-1} V_{\mathbf{o}}T_{\mathbf{o}}(\mathrm{x}_i, \mathrm{x}_{i+1}) = \prod_{\mathbf{o}\in\mathbf{O}} V_{\mathbf{o}}T_{\mathbf{o}}(\bar{x})\,,$$

where $V_{\mathbf{o}}T_{\mathbf{o}}(\bar{x})$ is the loss factor corresponding to object $\mathbf{o}$ over the entire path $\bar{x}$. This reordering is based on the finiteness of $\mathbf{O}$ and circumvents the non-multiplicativity of $1 - VT$, allowing us to take its complementary after loss factors have been accumulated over the path. However, the complementary is only taken among the set of casters $\mathbf{C}$; this defines the measurement contribution function for the shadow layer of $\mathbf{C}$:

$$\begin{aligned} f_{\mathbf{C},j}(\bar{x}) = {} & L_e(\mathrm{x}_0, \mathrm{x}_1)W_e^{(j)}(\mathrm{x}_{k-1}, \mathrm{x}_k)G(\mathrm{x}_0, \mathrm{x}_1) \\ & \cdot \prod_{i=1}^{k-1} F(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1})G(\mathrm{x}_i, \mathrm{x}_{i+1}) \\ & \cdot \prod_{\mathbf{o}\in\mathbf{O}\setminus\mathbf{C}} V_{\mathbf{o}}T_{\mathbf{o}}(\bar{x}) \cdot \prod_{\mathbf{c}\in\mathbf{C}} (1 - V_{\mathbf{c}}T_{\mathbf{c}}(\bar{x}))\,. \end{aligned} \tag{5.7}$$

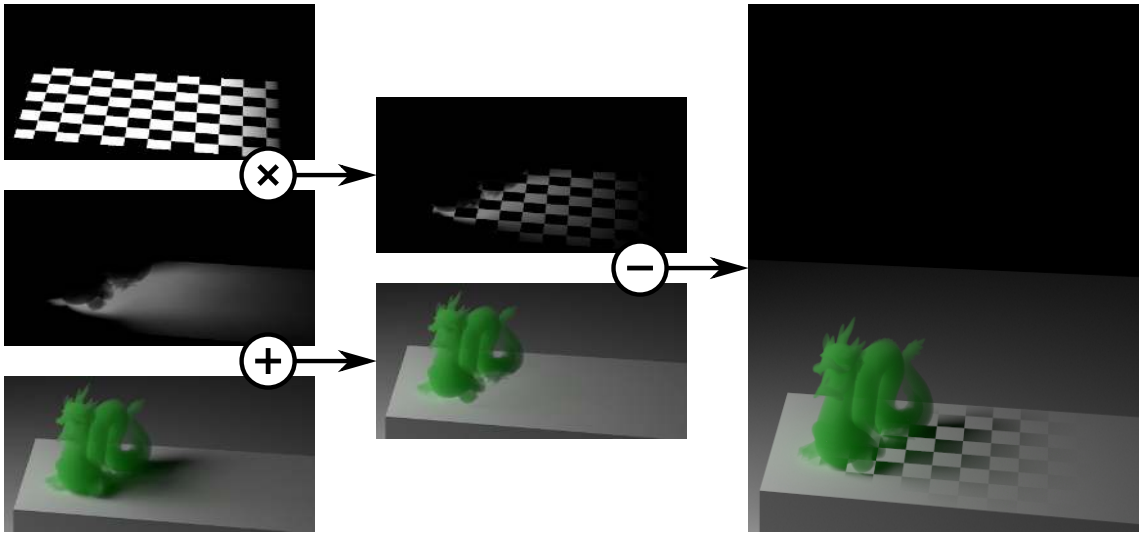


Figure 5.5: *The dragon is filled with an homogeneous medium and casts indirect shadow. We inlay a custom pattern over the pedestal using a simple compositing graph.*

Replacing Equation (5.7) into the path integral formulation and restricting the domain of integration to $\Omega \setminus \Omega_{\mathbf{C}}$ yields the shadow layer measurement at sensor position j:

$$S_{\mathbf{C},j} = \int_{\Omega \setminus \Omega_{\mathbf{C}}} f_{\mathbf{C},j}(\bar{x}) \, \mathrm{d}\mu(\bar{x}) \, . \tag{5.8}$$

The set of shadow casters $\mathbf{C}$ may be empty, in which case the path integral is simply that of the original image I. As detailed in Section 5.3, when $\mathbf{C} = \{\mathbf{c}\}$ with $\mathbf{c}$ a solid object, this formulation is equivalent to the definition of shadow layers from Chapter 4.

This new path integral naturally handles global illumination effects such as indirect shadows created by the occlusion of reflected light sources, as shown in the Dragon scene of Figure 5.5. Additionally, no restricting assumption is made on the optical properties of media in $\mathbf{C}$, such as homogeneity or phase function isotropy, making it general and compatible with realistic assets as demonstrated in Section 5.5.

5.3 • Equivalence for a single solid caster

We focus on the special case where the scene contains solid surfaces, and the set of casters is a single object: $\mathbf{C} = \{\mathbf{c}\}$. Omitting the dependence on the sensor position j and light path $\bar{x}$, the path integral formulation of Chapter 4 is

$$S_{\mathbf{c}} = \int_{\Omega_{\mathbf{c}}} f_{J,\mathbf{c}} \, \mathrm{d}\mu \tag{5.9}$$

with $f_{J,\mathbf{c}}$ the measurement contribution function where the BSDF of $\mathbf{c}$ is replaced at every interaction by $F_{\mathbf{c}}$, a straight transmission. $F_{\mathbf{c}}(\mathrm{x}_{i-1}, \mathrm{x}_i, \mathrm{x}_{i+1})$ is non-null only when x_{i-1}, x_i, and x_{i+1} are collinear and arranged in this order. Parameterized by incoming and outgoing directions ω_i and ω_o, it is written

$$F_{\mathbf{c}}(x, \omega_i, \omega_o) := \frac{\delta(\omega_i + \omega_o)}{|\mathrm{n}(x) \cdot \omega_i|} \, .$$

The path integrals in Equations (5.9) and (5.8) differ by their integration domains, $\Omega_{\mathbf{c}}$ and $\Omega \setminus \Omega_{\mathbf{c}}$ respectively. In order to connect the two, we define a mapping $\varphi : \Omega_{\mathbf{c}} \to \Omega \setminus \Omega_{\mathbf{c}}$ that removes all intersections between a path and $\mathcal{M}_{\mathbf{c}}$ the surface of $\mathbf{c}$. The mapping φ is highly surjective, as many paths in $\Omega_{\mathbf{c}}$ end up having the same image in $\Omega \setminus \Omega_{\mathbf{c}}$. We can reconstruct $\Omega_{\mathbf{c}}$ from all preimages of paths of length k in $\Omega \setminus \Omega_{\mathbf{c}}$:

$$\Omega_{\mathbf{c}} = \bigsqcup_{k=1}^{+\infty} \varphi^{-1}(\Omega \setminus \Omega_{\mathbf{c}} \cap \Omega_k)\,.$$

Proof:

$\subset$ Let $\bar{x} \in \Omega_{\mathbf{c}}$ that maps to $\bar{y} = \varphi(\bar{x})$, where $\bar{y}$ has length k; then $\bar{x} \in \varphi^{-1}(\Omega \setminus \Omega_{\mathbf{c}} \cap \Omega_k)$.

$\supset$ Each $\varphi^{-1}(\Omega \setminus \Omega_{\mathbf{c}} \cap \Omega_k) \subset \Omega_{\mathbf{c}}$, and the inclusion holds for the countable union over k.

$\sqcup$ Applying the preimage conserves set intersections, which are empty here.

From this observation, Equation (5.9) becomes

$$\int_{\Omega_{\mathbf{c}}} f_{J,\mathbf{c}} \,\mathrm{d}\mu = \sum_{k=1}^{+\infty} \int_{\varphi^{-1}(\Omega\setminus\Omega_{\mathbf{c}}\cap\Omega_k)} f_{J,\mathbf{c}} \,\mathrm{d}\mu\,,$$

and we also know that Equation (5.8) decomposes into

$$\int_{\Omega\setminus\Omega_{\mathbf{c}}} f_{\{\mathbf{c}\}} \,\mathrm{d}\mu = \sum_{k=1}^{+\infty} \int_{\Omega\setminus\Omega_{\mathbf{c}}\cap\Omega_k} f_{\{\mathbf{c}\}} \,\mathrm{d}\mu\,.$$

The equivalence of the path integrals will follow if we prove that for $k > 1$,

$$\int_{\varphi^{-1}(\Omega\setminus\Omega_{\mathbf{c}}\cap\Omega_k)} f_{J,\mathbf{c}} \,\mathrm{d}\mu = \int_{\Omega\setminus\Omega_{\mathbf{c}}\cap\Omega_k} f_{\{\mathbf{c}\}} \,\mathrm{d}\mu\,. \tag{5.10}$$

For simplicity, we assume that emitted radiance L_e and importance W_e are null over $\mathcal{M}_{\mathbf{c}}$. Let us focus on the case $k = 1$, and consider paths of length 2 in $\varphi^{-1}(\Omega \setminus \Omega_{\mathbf{c}} \cap \Omega_1)$ that encounter $\mathcal{M}_{\mathbf{c}}$ on their second vertex. They have the form $\bar{x} = \mathrm{x}_0\,\mathrm{x}_1\,\mathrm{x}_2$, where $\mathrm{x}_1 \in \mathcal{M}_{\mathbf{c}}$. The behavior of $F_{\mathbf{c}}$ strongly constrains the shape of paths that bring a non-null contribution to the integral. When we write the measurement contribution function

$$f_{J,\mathbf{c}}(\bar{x}) = L_e(\mathrm{x}_0,\mathrm{x}_1)GV(\mathrm{x}_0,\mathrm{x}_1)F_{\mathbf{c}}(\mathrm{x}_0,\mathrm{x}_1,\mathrm{x}_2)GV(\mathrm{x}_1,\mathrm{x}_2)W_e(\mathrm{x}_1,\mathrm{x}_2)$$

where $GV(x,y) = G(x,y)V(x,y)$, the collinearity of vertices x_0, x_1, and x_2 implies that the measurement does not change if we replace

$$L_e(\mathrm{x}_0,\mathrm{x}_1) = L_e(\mathrm{x}_0,\mathrm{x}_2) \quad \text{and} \quad W_e(\mathrm{x}_1,\mathrm{x}_2) = W_e(\mathrm{x}_0,\mathrm{x}_2)\,. \tag{5.11}$$

Also, Figure 5.6 illustrates that the presence of $\mathcal{M}_{\mathbf{c}}$ does not affect the change of area relative to projected solid angle, given by $GV(\mathrm{x}_i,\mathrm{x}_{i+1}) = \mathrm{d}\omega^{\perp}_{i+1} \,/\, \mathrm{d}A(\mathrm{x}_i)$. When we focus on the middle part of the path integral where x_1 covers $\mathcal{M}_{\mathbf{c}}$, simplifications occur:

$$\begin{aligned}
&\int_{\mathcal{M}_{\mathbf{c}}} GV(\mathrm{x}_0,\mathrm{x}_1)F_{\mathbf{c}}(\mathrm{x}_0,\mathrm{x}_1,\mathrm{x}_2)GV(\mathrm{x}_1,\mathrm{x}_2)\,\mathrm{d}A(\mathrm{x}_1)\\
&= \int_{\mathcal{M}_{\mathbf{c}}} \frac{\mathrm{d}\omega_1^{\perp}}{\mathrm{d}A(\mathrm{x}_0)} \frac{\delta(\omega_1-\omega_2)}{\mathrm{d}\omega_1^{\perp}\,/\,\mathrm{d}\omega_1} \frac{\mathrm{d}\omega_2^{\perp}}{\mathrm{d}A(\mathrm{x}_1)}\,\mathrm{d}A(\mathrm{x}_1)\\
&= \int_{\mathcal{S}^2} \frac{\mathrm{d}\omega_2^{\perp}}{\mathrm{d}A(\mathrm{x}_0)}\delta(\omega_1-\omega_2)\,\mathrm{d}\omega_1\\
&= G(\mathrm{x}_0,\mathrm{x}_2)\overline{V}^1_{\mathbf{c}}(\mathrm{x}_0,\mathrm{x}_2)\,.
\end{aligned} \tag{5.12}$$

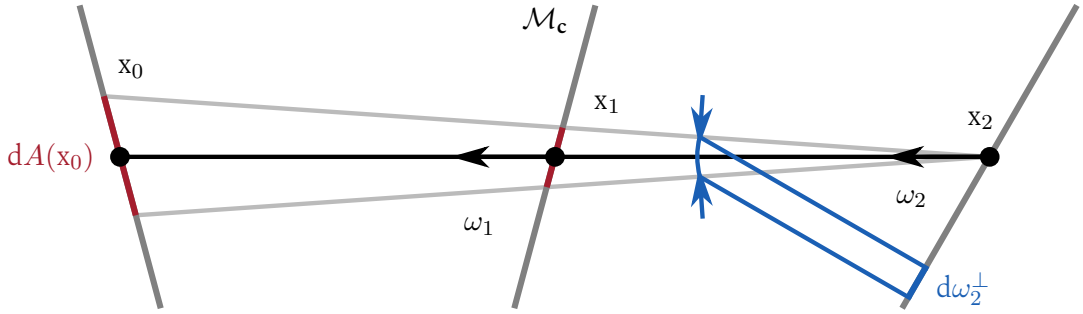


Figure 5.6: *The geometric factor* $\mathrm{d}\omega_2^\perp$ / $\mathrm{d}A(\mathrm{x}_0)$ *is not affected by* $\mathcal{M}_\mathbf{c}$*, except for visibility.*

Whereas the usual geometric factor $GV(x,y)$ rules the change of variables between the area measure $\mathrm{d}A$ on $\mathcal{M}$ the set of all surfaces in the scene and the projected solid angle measure $\mathrm{d}\omega^\perp$ on $\mathcal{S}^2$ the two-dimensional sphere of directions, the factor $G(x,y)\overline{V}_\mathbf{c}^m(x,y)$ we introduced results from the change of variables between $(\mathcal{M}_\mathbf{c})^m$ and $(\mathcal{S}^2)^m$ under the constraint of $F_\mathbf{c}$, with $m=1$ here. The modified visibility $\overline{V}_\mathbf{c}^m$ is defined as

$$\overline{V}_\mathbf{c}^m(x,y) = \begin{cases} 1 & \text{if there are exactly } m \text{ occlusions between } x \\ & \text{and } y\text{, and these occlusions are due to } \mathcal{M}_\mathbf{c}; \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

In summary, Equations (5.11) and (5.12) transform a path integral over vertices $\mathrm{x}_0,\ \mathrm{x}_1,\ \mathrm{x}_2$ where $\mathrm{x}_1 \in \mathcal{M}_\mathbf{c}$ and $\mathrm{x}_0, \mathrm{x}_2 \notin \mathcal{M}_\mathbf{c}$ into a path integral over vertices $\mathrm{x}_0,\ \mathrm{x}_2$. When considering paths of longer length in $\varphi^{-1}(\Omega \setminus \Omega_\mathbf{c} \cap \Omega_1)$, or when $k>1$, the simplifications described in these equations apply in chain: a measurement over $\varphi^{-1}(\Omega \setminus \Omega_\mathbf{c} \cap \Omega_k)$ always transforms into a measurement over $\Omega \setminus \Omega_\mathbf{c} \cap \Omega_k$. We now need to show that the measurement contribution function changes from $f_{J,\mathbf{c}}$ to $f_{\{\mathbf{c}\}}$ between the integrals of Equation (5.10). Indeed, we can rework the integration domain of the left integral without changing its value:

$$\varphi^{-1}(\Omega \setminus \Omega_\mathbf{c} \cap \Omega_k) \mapsto \bigsqcup_{(m_1,\ldots,m_k)\in(\mathbb{N}^k)^*} (\mathcal{M} \setminus \mathcal{M}_\mathbf{c}) \times (\mathcal{M}_\mathbf{c})^{m_1} \times \ldots \times (\mathcal{M}_\mathbf{c})^{m_k} \times (\mathcal{M} \setminus \mathcal{M}_\mathbf{c})$$

where m_i denotes the number of intersections that are removed between vertices x_{i-1} and x_i when φ is applied. The integral over $\varphi^{-1}(\Omega \setminus \Omega_\mathbf{c} \cap \Omega_k)$ becomes a sum of integrals that simplify using the rules of Equations (5.11) and (5.12), so that their domain turns into $\Omega \setminus \Omega_\mathbf{c} \cap \Omega_k$. As we add the different measurement contribution functions under the integral, a total visibility factors out of their sum:

$$\sum_{(m_1,\ldots,m_k)\in(\mathbb{N}^k)^*} \prod_{i=1}^{k} \overline{V}_\mathbf{c}^{m_i}(\mathrm{x}_{i-1},\mathrm{x}_i) \;= (1 - V_\mathbf{c}T_\mathbf{c}(\bar{x})) \prod_{\mathbf{o}\neq\mathbf{c}} V_\mathbf{o}T_\mathbf{o}(\bar{x})\,.$$

This is because, despite the summation on the left, only one tuple $(m_1,\ldots,m_k)$ may yield a non-null product of $\overline{V}_\mathbf{c}^{m_i}$ depending on the occlusion of the path. The last equality between visibility factors boils down to rewording the definition of $\overline{V}_\mathbf{c}^m$ based on Equation (5.13), which concludes the proof.

5.4 • Integration in a path tracing framework

Based on our path integral formulation, we present a path tracing algorithm that computes the original image and any number of generalized shadow layers in a single pass. Its only required parameter is a list of objects $\mathbf{c}_1, \ldots, \mathbf{c}_N$ for which shadow layers are requested by the user. We do not consider the internal representation of media (density grid, analytic expression, *etc.*) and only require that shadow casters are uniquely identified at each interaction they create along light paths.

5.4.1 • Key steps

The algorithm samples random light paths $\bar{x}$ and measures their contribution according to $f_{\mathbf{C}}$ for each of the 2^N possible unions of casters $\mathbf{C}$ that can be formed from the collection $\mathbf{c}_1, \ldots, \mathbf{c}_N$, as long as $\bar{x} \in \Omega \setminus \Omega_{\mathbf{C}}$. In order to compute the complementaries $1 - V_{\mathbf{c}_i} T_{\mathbf{c}_i}$ in each shadow layer $S_{\mathbf{C}}$ containing $\mathbf{c}_i$, we need to accumulate N loss factors during rendering. This is achieved through the modification of two key steps: the construction of the prefix path starting from the camera, and direct light gathering at vertices.

Path construction

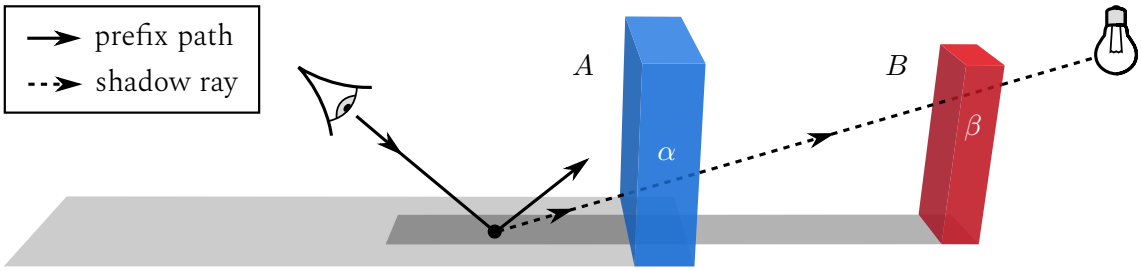
Indirect shadows are created when the light emitted from a source is scattered at least once, and then occluded before it reaches the camera. Because a path tracer follows the inverse direction of light, changing the way the prefix path is built is necessary to account for indirect shadows. Our path tracer may ignore the first interaction with a caster $\mathbf{c}_i$, and search for the amount of indirect shadow it creates. Specifically, there are two possible outcomes for each interaction with a caster:

- The event is discarded, and the path does not interact with the caster anymore: $\mathbf{c}_i$ is traversed without scattering. Its loss factor $V_{\mathbf{c}_i} T_{\mathbf{c}_i}$ is set to 0, as an interaction with the object would have normally prevented light from going straight through. As the path belongs to $\Omega \setminus \Omega_{\{\mathbf{c}_i\}}$, it contributes to shadow layers according to Equation (5.7).
- The event occurs normally, and thus the impact of $\mathbf{c}_i$ on propagation is acknowledged. If $\mathbf{c}_i$ is encountered again, the algorithm proceeds with the event. The path now belongs to $\Omega_{\{\mathbf{c}_i\}}$ and it will not contribute to shadow layers $S_{\mathbf{C}}$ for which $\mathbf{c}_i \in \mathbf{C}$.

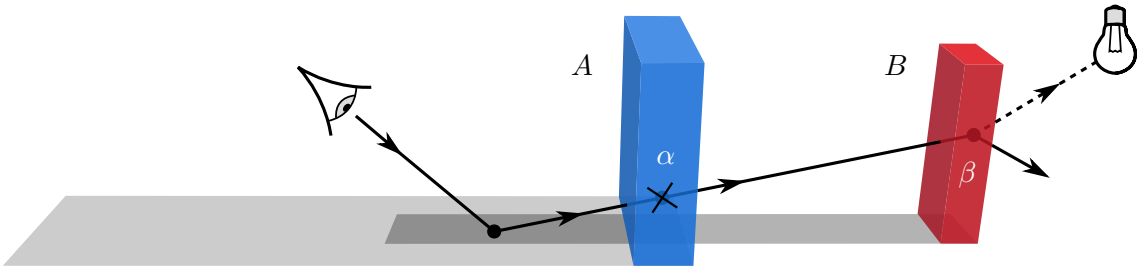
These outcomes are chosen at random, following the one-sample model with respective probabilities γ and $1 - \gamma$. By default, we set $\gamma = 1/2$.

Direct light gathering

At each vertex of the prefix path, the algorithm sends a shadow ray toward a light source. Changing the behavior of this shadow ray allows us to pick up shadow at the same time as radiance. As illustrated in Figure 5.7, a modified shadow ray keeps track of separate loss factors $V_{\mathbf{c}_i} T_{\mathbf{c}_i}$ for each traversed caster $\mathbf{c}_i$ that was never encountered during propagation. This implies that solid occluders may be ignored when testing visibility with the light if they are marked as casters. The radiance emitted by the source then contributes to each shadow layer according to Equation (5.7), where we combine the loss factors stored in the shadow ray with those of the prefix path. Two examples of paths generated by the algorithm are given in Figure 5.7, based on the setup of Figure 5.1.

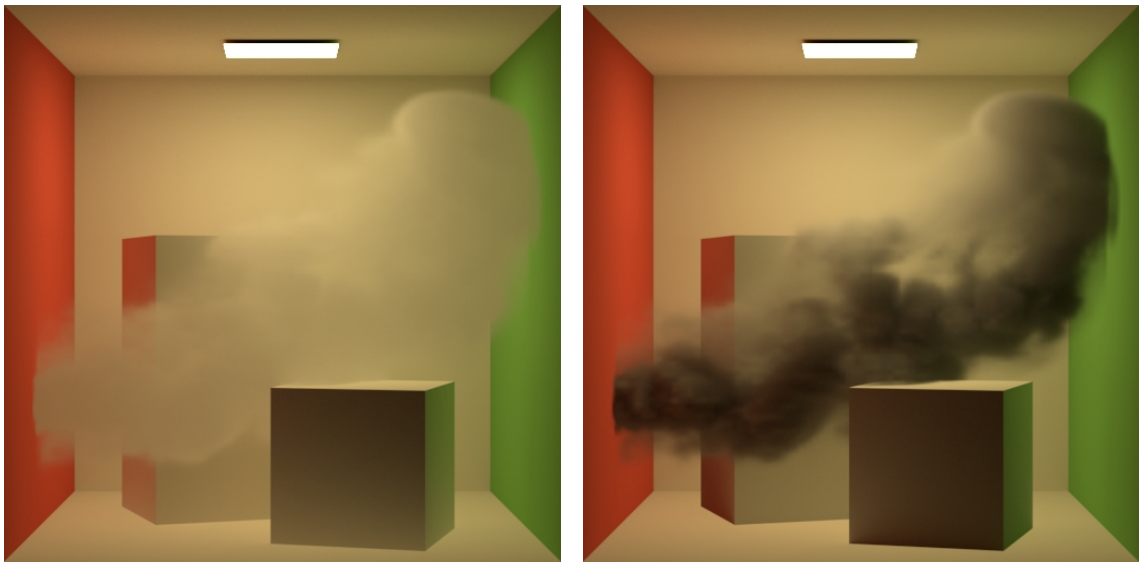


The shadow ray contributes to the original image I with factor $\alpha\beta$, and to the mutual shadow layer $S_{\{A,B\}}$ of the two media with factor $1 - \alpha\beta$. All radiance from the source is thus recovered, inside different layers.



A volumetric interaction sampled in A is ignored, and the path continues until another one in B. The next shadow ray does not contribute radiance to I, but indirect shadow to $S_{\{A\}}$ with factor $(1 - \alpha)\beta = \beta$.

Figure 5.7: *Our volumetric path tracing algorithm executed in the setup of Figure 5.1. It accounts for mutual (top) and indirect (bottom) shadows from the two participating media.*



Removal with self-shadowing *Removal without self-shadowing*

Figure 5.8: *Left: shadow removal for the smoke and the two boxes, including self-shadowing. Right: the same setup, but ignoring self-shadowing for all casters.*

5.4.2 • Additional parameters

Beyond the two key steps required to obtain shadow, we describe various refinements to the design of our path tracing algorithm that bring more control over the content of shadow layers and the performance of rendering.

We enable two artistic parameters that were discussed in the implementation for solid objects (Section 4.4). The user is able to define custom catchers, on which shadow should be measured. If a shadow caster is also a catcher, we give the option to toggle off self-shadowing. The result of self-shadowing is especially pronounced when dealing with participating media, as it contains the result of absorption by the medium. Figure 5.8 shows the difference between the two on the Cornell Box scene.

Moreover, the performance parameters that were discussed in Section 4.6.1 are also proposed to the user. The maximum number of scattering events that can be discarded along a path is set to twice the maximum path depth by default, but can be freely changed. The probability γ is also exposed as a parameter, offering a trade-off between indirect shadow sampling in shadow layers, and indirect light sampling in the original image.

Maximum cardinal

By default, the prototype renders all $2^N - 1$ shadow layers that can exist in the presence of N different shadow casters. This exponential complexity is particularly problematic, not only because it slows down rendering in practice, but because it invalidates the motivation of our approach. Indeed, we discarded the subtraction method where shadow layers are obtained from two alternative renders, because it requires twice more computations. Yet, doubling the number of renders is equivalent to considering one more shadow caster under the current complexity: $2 \times 2^N = 2^{N+1}$.

At the end of the algorithm, we compute the second moment of the XYZ color space luminance Y in each of the layers, to estimate the energy they contain; we discard those where it falls under a user-controlled threshold. This convenient feature prevents the export of too many images with negligible contribution, but in itself does not reduce the exponential complexity that comes along with our formulation.

However, using this thresholding on various scenes brought us to the conclusion that shadow layers for which the set of casters $\mathbf{C}$ is large are usually discarded, as multiple occlusions involving all of their elements are not likely to happen in the scene. For this reason, we propose a last parameter that controls the maximum cardinal of the considered sets of shadow casters. We have seen that considering all possible interactions between k casters among N total yields $\binom{N}{k}$ layers (Section 4.7). When k is bounded by a constant K_m, we obtain the following asymptotic behavior for the total number of layers:

$$\sum_{k=0}^{K_m} \binom{N}{k} = \sum_{k=0}^{K_m} \frac{N!}{k!(N-k)!} = \sum_{k=0}^{K_m} \frac{N(N-1)\dots(N-(k-1))}{k!} = \sum_{k=0}^{K_m} \mathcal{O}(N^k) = \mathcal{O}(N^{K_m})$$

The total number of layers being closely linked to the complexity of the algorithm, we can expect drastically better performance compared to the original $\mathcal{O}(2^N)$. The speedup due to parameter K_m is quantified precisely in the following section.

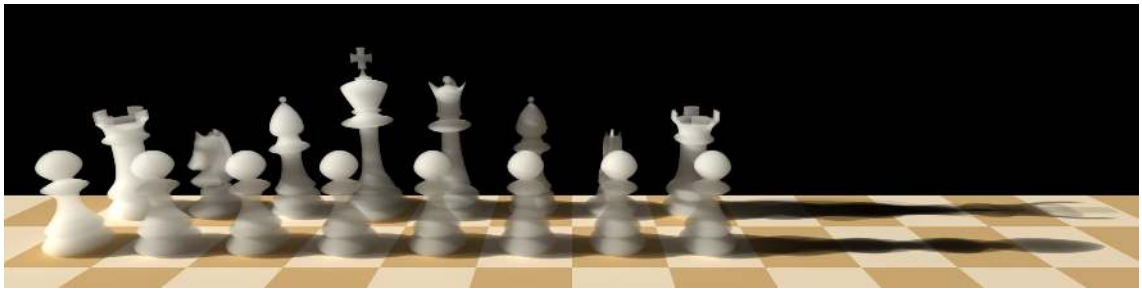
5.5 • Results and performance

Our prototype shadow integrator is based on the volumetric path tracer of pbrt-v3 [Pha16]. We ran our measurements on an Intel Xeon E5-2630 v4 processor with 20 threads at 2.20 GHz and 64 GB RAM, and display the results in Table 5.1. N is the number of shadow casters in the scene. When $N = 0$, the standard volumetric path tracer is used to render only the original image and when $N > 0$, our algorithm renders 2^N layers: the original image and $2^N - 1$ shadow layers. We include in Table 5.1 the rendering times, Root-Mean-Square Error (RMSE) between original images, and Zero Radiance Paths (ZRP) percentage for the different figures of this chapter.

Turning on the export of shadow layers incurs a performance overhead in rendering time and memory usage, attributable to several factors. First, while managing additional images has a predictable memory footprint, it involves running numerous 2-dimensional loops to apply reconstruction filters around the samples. Second, and most observable in the measurements, our algorithm accumulates the product of N loss factors to form 2^N factors over every sampled path. Indeed when incoming radiance L is picked up, each of the 2^N layers i receives a contribution $\alpha_i L$ where α_i is the final loss factor over the light path. The factor α_i is the product of all object loss factors $V_{\mathbf{o}}T_{\mathbf{o}}$ or their complementary $1 - V_{\mathbf{o}}T_{\mathbf{o}}$, according to Equation (5.7). The overhead remains consistently under 15% in our experiments except for the Chess scene, which is designed for paths to encounter many casters. Performance in this setup is compared separately in Figure 5.9 by increasing the number of shadow casters from $N = 0$ (standard pbrt-v3 integration) to 10 and measuring rendering times. By default, the resulting data exhibits the 2^N behavior expected expected when accounting for all possible interactions between N casters. However, we also show that the maximum cardinal parameter K_m greatly reduces complexity in practice.

Scene	N	Layers	Samples	Time	RMSE $\times 10^3$	ZRP
Cornell Box (Figure 5.4)	0	1	1024	8’ 08”	1.2833	56%
	1	2	1024	8’ 23”	1.3454	28%
	3	8	1024	9’ 13”	1.7062	21%
Dragon (Figure 5.5)	0	1	4096	17’23”	7.2070	92%
	1	2	4096	17’58”	7.2790	91%
Beach (Figure 5.10)	0	1	256	67’44”	1.6333	71%
	3	8	256	68’51”	1.6350	43%
Manhole (Figure 5.11)	0	1	512	9’57”	2.5707	30%
	1	2	512	10’12”	2.5732	18%

Table 5.1: *Performance comparison for N shadow casters between a standard path tracer (N = 0) and our shadow integrator (N > 0). Rendering times increase with an overhead consistently under 15%; the RMSE is computed for the original images, compared to a converged reference. The decrease in Zero Radiance Paths (ZRP) with more layers supports our single pass approach.*



The Chess scene involves numerous occlusions by different shadow casters along light paths.

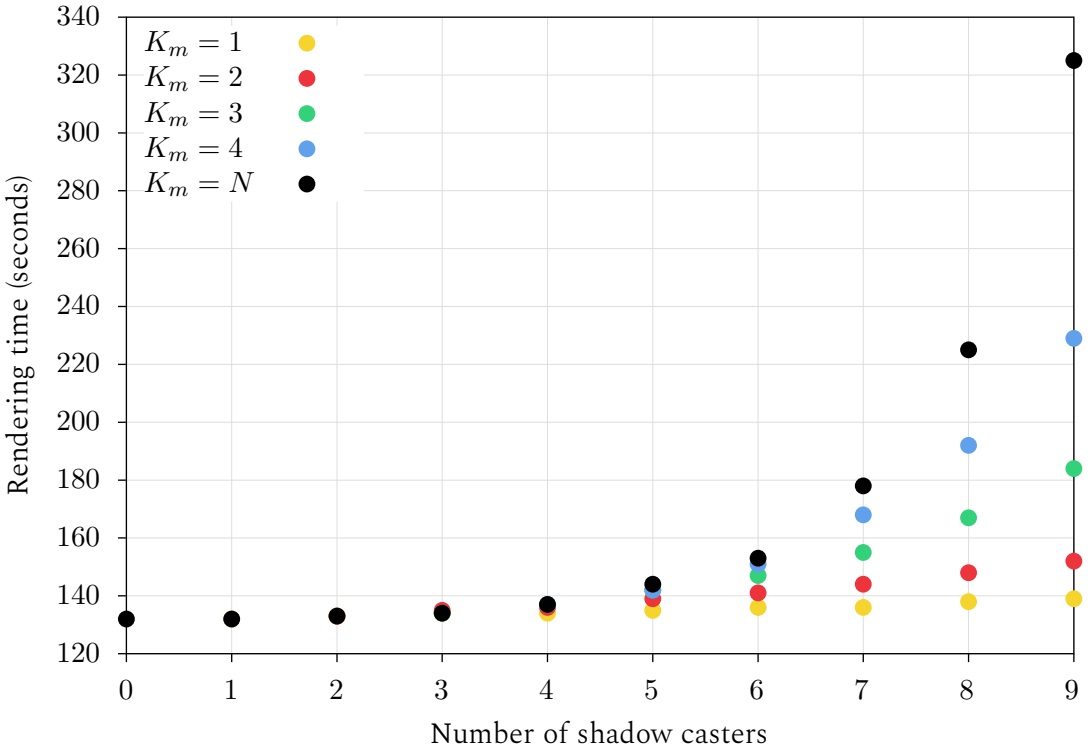


Figure 5.9: *Evolution of the rendering time in the Chess scene (top) at 1024 samples per pixel. By default, the algorithm renders shadow layers for all possible combinations of N casters, with complexity $\mathcal{O}(2^N)$. Setting the maximum cardinal of combinations to K_m reduces the complexity to $\mathcal{O}(N^{K_m})$, but intricate shadows created by more than K_m occluders along a ray are lost.*



Original image



Shadow layers of the three clouds, each considered separately (no coupling)



Composite result with the performed edits shown in insets

Figure 5.10: *Top: in the Beach scene, three instances of Disney's Cloud Data Set are disposed above the scenery. Middle: the shadow layers of all three and their possible unions are exported, but we only use the layers without coupling to perform the edits. Top, left: shape simplification using a median filter; center: the shadow's intensity is modulated by a gradient; right: a custom painting of the Stanford Bunny replaces the shadow ratio of the last cloud.*

Compared to a standard path tracer, convergence at each pixel is affected: the same number of samples is now distributed among a number of different layers, meaning that the original image does not receive as many contributions. This is measured by the RMSE, computed against a converged reference image using at least 16 times the sampling budget.

The Beach scene is a rendering of the Moana Island Scene featuring around 50 million unique triangles and 20 million parametric curves (see Figure 5.10). We disposed three distinct instances of Walt Disney Animation Studios' Cloud Data Set at half resolution above the beach, for a total of 9.2 GB uncompressed density data. Measurements show that our algorithm has a negligible impact on this production-grade scene, where the overall complexity of surfaces and volumes prevails: the overhead in rendering time is under 2%, while the convergence of the original image is barely affected.

In the Cornell Box scene, we begin by rendering a single shadow layer for the smoke, and follow with a total of 8 layers by also considering the two boxes. While the number of zero radiance paths systematically decreases, we notice a deterioration of performance in rendering time and convergence. Whereas rendering time is mainly affected by the number of layers, the worse convergence is due to light reflecting on walls and creating strong indirect shadows that span a large area of the image. In turn, many generated paths end up measuring indirect shadows, and do not contribute to the original image anymore.

Our approach is compatible with animated scenes, as demonstrated by the evolution of the shadow ratio $I/(I+S)$ in the Manhole sequence of Figure 5.11

While our single pass implementation requires more computations, it is legitimated by the systematic decrease in zero radiance paths. Building paths is costly as it involves numerous intersection tests to simulate the propagation of rays. By applying a different measurement contribution functions for each rendered layer to every single path, we fully leverage the cost of ray-scene intersections. In many instances, we also reuse otherwise lost information: even when it is exponential, transmittance may be null in practice when delta tracking [Nov18] is used for its estimation. In this case, our algorithm picks up shadow where a standard path tracer would return no radiance.



Figure 5.11: *The frames 20, 30, and 40 of the Manhole scene (top row) show that shadow ratios (bottom row) behave coherently in the presence of animations.*

5.6 • Conclusion

We have begun this chapter by analyzing the exchanges of energy between a light ray and a participating medium. These exchanges are modeled by the radiative transfer equation that summarizes gains and losses from the medium, and naturally relates to the notion of impact. We were able to quantify the negative part of impact from absorption and out-scattering of photons along the ray using the complementary of transmittance.

However, our understanding of negative volumetric impact remained local and only accounted for a single participating medium. When trying to generalize our observations to entire light paths, we were limited by the non-multiplicativity of the complementary of transmittance. To take the complementary of transmittance only after it has been accumulated over all rays of a path, we resorted to the path integral formulation for volumetric light transport. We were then able to extend the measure of shadow to arbitrary sets of both volumetric and solid casters. We proved that this new path integral is equivalent to that of Chapter 4 for a single solid shadow caster, and is thus a proper generalization.

Based on this path integral formulation of shadow, we proposed a path tracing algorithm that renders the original image and the $2^N - 1$ shadow layers that exist between N casters. While this complexity becomes prohibitive when N is too large, we have shown that it can be reduced to $\mathcal{O}(N^{K_m})$ if we only account for at most K_m different occluders along a path. We then tested our prototype on several challenging scenes, and displayed various compositing edits made possible by our approach.

Chapter 6

Conclusion

We have presented shadow layers, a new approach to efficiently capture the amount of light lost in a 3D scene due to any number of occluding objects under global illumination.

6.1 • Summary

We began this dissertation by exposing our motivation for light and shadow editing in Chapter 1, focusing on computer-generated imagery. Chapter 2 gave an overview of the existing methods that apply to synthetic imagery. Only light path expressions, discussed in Chapter 3, allow real-time editing of global illumination lighting features in the form of layers. They do not capture shadows however, and Chapter 4 has shown that the common techniques to extract shadow inside separate layers are prone to errors. Existing approaches are limited to the extraction of shadows from direct light sources, and overestimate illumination if an object's shadow is removed.

Our first definition of shadow layers overcomes these limitations, and has an intrinsic physical meaning: a shadow layer contains the amount of light lost on surfaces of the scene due to a solid occluder. This definition is based on a path integral formulation, and thus amenable to Monte Carlo integration. Indeed, we have implemented a path tracing algorithm that renders the original image and any number of shadow layers in a single pass. After analyzing its performance, we concluded on an overhead factor between 1.1 and 1.3, and quantified the convergence penalty on all exported images. We also showed that our single pass approach is justified by the systematic decrease in zero radiance paths with more shadow casters. This path tracer is implemented in pbrt-v3 and Arnold for Maya, and we discussed possible implementations with other popular integration schemes such as bidirectional path tracing, photon mapping, and Metropolis light transport.

While the original definition of shadow layers considers solid casters in isolation from one another, Chapter 5 generalizes the method to arbitrary sets of objects: this allows us to pick up shadows created after several occlusions by different casters. In addition, we extended the method to volumetric casters without any assumption on their physical properties. Our solution is thus compatible with production-grade scenes, as demonstrated by our generalized path tracer. While it exports an exponential number of layers by default, we complemented it with a simple parameter that reverts to a polynomial complexity.

6.2 • Future work

The contributions we have presented can be greatly improved upon, and we list several interesting avenues of development for the future.

Covariance in light path expressions

In Chapter 3, we have presented and reproduced the state of the art in light path expressions, that allow the clustering of lights paths according to their propagation history. We reckon that a more precise description of this history could improve light path expressions. We have investigated the theory of covariance tracing [Bel12] to provide a lightweight description of frequency changes in the light field during propagation. It quantifies the effect of surface curvature, sharpness of the BSDF, or occlusion for instance. While our prototype implementation did not outperform the precision of object identifiers when creating clusters, we believe in a fruitful interaction between the two methods.

Export of deep images

We have seen in Section 4.8 that compositing artists resort to deep images to properly rework renders that contain translucent surfaces or participating media. Deep images provide additional information in image space as every pixel keeps track of partial occlusions along primary rays instead of storing only the final color on the sensor. As deep images simplify the compositing of participating media, generalized shadow layers could benefit from such a representation. Our pbrt-v3 prototype implementation does not export deep images; their addition is orthogonal to our work, but possible.

Automatic selection of layers

One concern of Section 5.4 was that the number of possible interactions between N casters leads to 2^N layers total, and thus to a exponential complexity. We have shown that limiting the maximum cardinal of caster sets to K_m brings down the complexity to $\mathcal{O}(N^{K_m})$, but this user parameter may discard valuable layers. We would like to investigate a solution where an initial bootstrap phase is added to estimate the contribution of each layer, and remove those containing too little energy before the main rendering pass begins.

Generation of a training dataset

We believe shadow layers are also useful outside of the compositing pipeline. Specifically, our method could apply to deep learning approaches where networks are trained to infer or remove shadow in natural images [Phi19, Phi21, Nic20]. Compared to binary masks, our representation is linear and should better perform in linear operations such as convolution. Whereas shadow layers do not directly provide a segmentation as masks do, we reckon that they could generate more accurate models for tasks such as shadow removal or relighting where no segmentation is required.

Bibliography

[Ada01] A. Adamson and V. Jenson, 2001. *Shrek*. Dreamworks Animation, Dreamworks Pictures. ↑ 3

[Anj06] K.-i. Anjyo, S. Wemler, and W. Baxter, 2006. Tweakable Light and Shade for Cartoon Animation. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '06, pages 133–139. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1124728.1124750. ↑ 14

[Arv93] J. Arvo, 1993. Transfer Equations in Global Illumination. In *Global Illumination, SIGGRAPH '93 Course Notes*. ↑ 65

[Aut21a] Autodesk, 2021. *Closures - Arnoldpedia - Arnold Renderer*. https://docs.arnoldrenderer.com/display/A5ARP/Closures. Accessed 2021-09-01. ↑ 58, 59

[Aut21b] Autodesk, 2021. *Shadow Matte - Arnold for Maya User Guide - Arnold Renderer*. https://docs.arnoldrenderer.com/display/A5AFMUG/Shadow+Matte. Accessed 2021-09-01. ↑ 39

[Ban20] S. P. Bangaru, T.-M. Li, and F. Durand, November 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39 (6). http://dx.doi.org/10.1145/3414685.3417833. ↑ 16

[Bel12] L. Belcour, 2012. *A Frequency Analysis of Light Transport*. Ph.D. thesis. ↑ 82

[Bir04] B. Bird and J. Walker, 2004. *The Incredibles*. Pixar Animation Studios, Walt Disney Pictures. ↑ 1

[Bir07] B. Bird and B. Lewis, 2007. *Ratatouille*. Pixar Animation Studios, Walt Disney Pictures. ↑ 3

[Bon17] N. Bonneel, B. Kovacs, S. Paris, and K. Bala, 2017. Intrinsic Decompositions for Image Editing. *Computer Graphics Forum (Eurographics State of the Art Reports 2017)* 36 (2). ↑ 19

[Bou11] A. Bousseau, E. Chapoulie, R. Ramamoorthi, and M. Agrawala, 2011. Optimizing Environment Maps for Material Depiction. In *Proceedings of the Twenty-second Eurographics Conference on Rendering*, EGSR '11, pages 1171–1180. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. http://dx.doi.org/10.1111/j.1467-8659.2011.01975.x. ↑ 14

[BR20] I. Baeza Rojo, M. Gross, and T. Günther, 2020. Controllable Caustic Animation Using Vector Fields. In A. Wilkie and F. Banterle, editors, *Eurographics 2020 - Short Papers*. The Eurographics Association. http://dx.doi.org/10.2312/egs.20201005. ↑ 12

[Bre20] C. Brejon, 2020. *The CG Cinematography Book*. https://chrisbrejon.com/cg-cinematography/chapter-9-compositing/#light-path-expression. Accessed 2021-09-01. ↑ 2, 20, 30

[Cas00] R. Casati, 2000. *La scoperta dell'ombra da Platone a Galileo : la storia di un enigma che ha affascinato le grandi menti dell'umanità*. Mondadori. ↑ 5

[Cas19] R. Casati and P. Cavanagh, 2019. *The Visual World of Shadows*. MIT Press. ↑ 5

[Cha60] S. Chandrasekhar, 1960. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications. ↑ 65

[Cha21] Chaos Group, 2021. *Raw Shadow - V-ray 5 for Maya*. https://docs.chaosgroup.com/display/VMAYA/Raw+Shadow. Accessed 2021-09-01. ↑ 51

[Chu03] Y.-Y. Chuang, D. B. Goldman, B. Curless, D. H. Salesin, and R. Szeliski, 2003. Shadow Matting and Compositing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 494–500. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1201775.882298. ↑ 19

[Coo98] B. Cook, T. Bancroft, and P. Coats, 1998. *Mulan*. Walt Disney Pictures, Walt Disney Feature Animation, Walt Disney Studios Motion Pictures. ↑ 4

[Cos99] A. C. Costa, A. A. Sousa, and F. Nunes Ferreira, 1999. Lighting Design: A Goal Based Approach Using Optimisation. In D. Lischinski and G. W. Larson, editors, *Rendering Techniques' 99*, pages 317–328. Springer Vienna, Vienna. ↑ 15

[Dac07] C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand, July 2007. Implicit Visibility and Antiradiance for Interactive Global Illumination. *ACM Trans. Graph.* 26 (3). http://dx.doi.org/10.1145/1276377.1276453. ↑ 41

[DeC07] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz, 2007. Stylized Shadows. In *Proc. 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, pages 77–83. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1274871.1274884. ↑ 20

[Dee88] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt, June 1988. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. *SIGGRAPH Comput. Graph.* 22 (4), pages 21–30. http://dx.doi.org/10.1145/378456.378468. ↑ 33

[Des89] R.-C. Desrichard, 1989. *Le vocabulaire de la vie à la ferme dans l'ancien archiprêtré de Souvigny*. Ph.D. thesis, Clermont-Ferrand 2.

[Des18] F. Desrichard and D. Vanderhaeghe, November 2018. Analysis of Path Space for Layered Light Editing. *Journées Françaises d'Informatique Graphique* . ↑ 8

[Des19] F. Desrichard, D. Vanderhaeghe, and M. Paulin, July 2019. Global Illumination Shadow Layers. *Computer Graphics Forum* 38 (4), pages 183–191. http://dx.doi.org/10.1111/cgf.13781. ↑ 8

[Des21] F. Desrichard, D. Vanderhaeghe, and M. Paulin, December 2021. Shadow Layers for Participating Media. *Computer Graphics Forum* 40. http://dx.doi.org/10.1111/cgf.14429. ↑ 8

[Doc15] P. Docter and J. Rivera, 2015. *Inside Out*. Pixar Animation Studios, Walt Disney Studios Motion Pictures. ↑ 2

[Duc15] S. Duchêne, C. Riant, G. Chaurasia, J. L. Moreno, P.-Y. Laffont, S. Popov, A. Bousseau, and G. Drettakis, November 2015. Multiview Intrinsic Images of Outdoors Scenes with an Application to Relighting. *ACM Trans. Graph.* 34 (5). http://dx.doi.org/10.1145/2756549. ↑ 19

[Fas19] L. Fascione, J. Hanika, D. Heckenberg, C. Kulla, M. Droske, and J. Schwarzhaupt, 2019. Path Tracing in Production: Part 1: Modern Path Tracing. In *ACM SIGGRAPH 2019 Courses*, SIGGRAPH '19. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/3305366.3328079. ↑ 29

[Fav17] J.-D. Favreau, F. Lafarge, and A. Bousseau, November 2017. Photo2clipart: Image Abstraction and Vectorization Using Layered Linear Gradients. *ACM Trans. Graph.* 36 (6). http://dx.doi.org/10.1145/3130800.3130888. ↑ 19

[Gka15] A. Gkaravelis, G. Papaioannou, and K. Kalampokis, 2015. Inverse Light Design for High-Occlusion Environments. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications*, GRAPP 2015, pages 26–34. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT. http://dx.doi.org/10.5220/0005291400260034. ↑ 15

[Gor84] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, 1984. Modeling the Interaction of Light between Diffuse Surfaces. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 213–222. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/800031.808601. ↑ 10

[Gri16] L. Gritz, 2016. *OSL Light Path Expressions*. https://github.com/imageworks/OpenShadingLanguage/wiki/OSL-Light-Path-Expressions. Accessed 2021-09-01. ↑ 31

[Gün16] T. Günther, K. Rohmer, C. Rössl, T. Grosch, and H. Theisel, 2016. Stylized Caustics: Progressive Rendering of Animated Caustics. *Computer Graphics Forum* 35 (2), pages 243–252. http://dx.doi.org/10.1111/cgf.12827. ↑ 15

[Gur10] J. Gurney, 2010. *Color and Light: A Guide for the Realist Painter*. James Gurney Art Series. Andrews McMeel Publishing. ↑ 2

[Gut08] D. Gutierrez, F. J. Seron, J. Lopez-Moreno, M. P. Sanchez, J. Fandos, and E. Reinhard, 2008. Depicting Procedural Caustics in Single Images. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1457515.1409073. ↑ 15

[Hac09] T. Hachisuka and H. W. Jensen, 2009. Stochastic Progressive Photon Mapping. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1661412.1618487. ↑ 12, 58

[Han90] P. Hanrahan and P. Haeberli, September 1990. Direct WYSIWYG Painting and Texturing on 3D Shapes. *SIGGRAPH Comput. Graph.* 24 (4), pages 215–223. http://dx.doi.org/10.1145/97880.97903. ↑ 12

[Hec90] P. S. Heckbert, 1990. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 145–154. ACM, New York, NY, USA. http://dx.doi.org/10.1145/97879.97895. ↑ 30

[Hec14] M. Hecher, M. Bernhard, O. Mattausch, D. Scherzer, and M. Wimmer, July 2014. A Comparative Perceptual Study of Soft-Shadow Algorithms. *ACM Trans. Appl. Percept.* 11 (2). http://dx.doi.org/10.1145/2620029. ↑ 5

[Ige99] H. Igehy, 1999. Tracing Ray Differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 179–186. ACM Press / Addison-Wesley Publishing Co., USA. http://dx.doi.org/10.1145/311535.311555. ↑ 15

[Kah96] J. Kahrs, S. Calahan, D. Carson, and S. Poster, 1996. Pixel Cinematography: A Lighting Approach for Computer Graphics. In *ACM SIGGRAPH 1996 Courses*. Association for Computing Machinery. ↑ 1

[Kaj86] J. T. Kajiya, 1986. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/15922.15902. ↑ 25, 29

[Kas99] R. Kasrai, F. A. A. Kingdom, and T. M. Peters, 1999. The Perception of Transparency in Medical Images. In C. Taylor and A. Colchester, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99*, pages 726–733. Springer Berlin Heidelberg, Berlin, Heidelberg. ↑ 18

[Kaw93] J. K. Kawai, J. S. Painter, and M. F. Cohen, September 1993. Radioptimization - Goal Based Rendering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 147–154. Association

for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/166117.166136. ↑ 14

[Ker97] D. Kersten, P. Mamassian, and D. C. Knill, 1997. Moving Cast Shadows Induce Apparent Motion in Depth. *Perception* pages 171–192. ↑ 5

[Ker09] W. B. Kerr and F. Pellacini, July 2009. Toward Evaluating Lighting Design Interface Paradigms for Novice Users. *ACM Trans. Graph.* 28 (3). http://dx.doi.org/10.1145/1531326.1531332. ↑ 10

[Ker10] W. B. Kerr, F. Pellacini, and J. D. Denning, 2010. Bendylights: Artistic Control of Direct Illumination by Curving Light Rays. In *Proceedings of the 21st Eurographics Conference on Rendering*, EGSR'10, pages 1451–1459. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. http://dx.doi.org/10.1111/j.1467-8659.2010.01742.x. ↑ 16

[Kle14] O. Klehm, I. Ihrke, H.-P. Seidel, and E. Eisemann, 2014. Property and Lighting Manipulations for Static Volume Stylization Using a Painting Metaphor. *IEEE Transactions on Visualization and Computer Graphics* 20 (7), pages 983–995. http://dx.doi.org/10.1109/TVCG.2014.13. ↑ 12

[Laf13] P.-Y. Laffont, A. Bousseau, and G. Drettakis, February 2013. Rich Intrinsic Image Decomposition of Outdoor Scenes from Multiple Views. *IEEE Transactions on Visualization and Computer Graphics* 19 (2), pages 210–224. http://dx.doi.org/10.1109/TVCG.2012.112. ↑ 19

[Law99] C. Lawrence, J. Zhou, and A. Tits, February 1999. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. *Technical report TR-94-16r1, University of Maryland, College Park* . ↑ 10

[Li18] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, December 2018. Differentiable Monte Carlo Ray Tracing Through Edge Sampling. *ACM Trans. Graph.* 37 (6), pages 222:1–222:11. http://dx.doi.org/10.1145/3272127.3275109. ↑ 16

[Lle19] P. Llerena, 2019. *Among other positions, Philippe Llerena has taught lighting and compositing at Isart Digital School.* By videoconference. ↑ 39

[Lor18] P. Lord, B. Persichetti, P. Ramsey, and R. Rothman, 2018. *Spider-Man: Into the Spider-Verse*. Marvel Comics, Columbia Pictures, Marvel Entertainment, Sony Pictures Animation, Sony Pictures Releasing. ↑ 6

[Lou19] G. Loubet, N. Holzschuch, and W. Jakob, December 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38 (6). http://dx.doi.org/10.1145/3355089.3356510. ↑ 16

[Mat13] O. Mattausch, T. Igarashi, and M. Wimmer, May 2013. Freeform Shadow Boundary Editing. *Computer Graphics Forum* 32, pages 175–184. http://dx.doi.org/10.1111/cgf.12037. ↑ 13

[McK17] C. McKay, D. Lin, P. Lord, C. Miller, and R. Lee, 2017. *The Lego Batman Movie*. Warner Animation Group, Warner Bros. Pictures. ↑ 2

[Mek21] A. Meka, M. Shafiei, M. Zollhoefer, C. Richardt, and C. Theobalt, January 2021. Real-time Global Illumination Decomposition of Videos. http://dx.doi.org/10.1145/3374753. ↑ 19

[Mit09] N. J. Mitra and M. Pauly, December 2009. Shadow Art. *ACM Trans. Graph.* 28 (5), pages 1–7. http://dx.doi.org/10.1145/1618452.1618502. ↑ 4

[ND20] M. Nimier-David, S. Speierer, B. Ruiz, and W. Jakob, July 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39 (4). http://dx.doi.org/10.1145/3386569.3392406. ↑ 16

[Ngu13] C. H. Nguyen, D. Scherzer, T. Ritschel, and H.-P. Seidel, May 2013. Material Editing in Complex Scenes by Surface Light Field Manipulation and Reflectance Optimization. *Computer Graphics Forum* 32 (2), pages 185–194. http://dx.doi.org/10.1111/cgf.12038. ↑ 11

[Nic20] B. Nicolet, J. Philip, and G. Drettakis, 2020. Repurposing a Relighting Network for Realistic Compositions of Captured Scenes. In *Symposium on Interactive 3D Graphics and Games*, I3D '20. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/3384382.3384523. ↑ 19, 82

[Nov18] J. Novák, I. Georgiev, J. Hanika, and W. Jarosz, May 2018. Monte Carlo Methods for Volumetric Light Transport Simulation. *Computer Graphics Forum* 37 (2), pages 551–576. http://dx.doi.org/10.1111/cgf.13383. ↑ 78

[Now11] D. Nowrouzezahrai, J. Johnson, A. Selle, D. Lacewell, M. Kaschalk, and W. Jarosz, July 2011. A Programmable System for Artistic Volumetric Lighting. *ACM Trans. Graph.* 30 (4), pages 29:1–29:8. http://dx.doi.org/10.1145/2010324.1964924. ↑ 17

[Obe08] J. Obert, J. Křivánek, F. Pellacini, D. Sýkora, and S. Pattanaik, 2008. iCheat: A Representation for Artistic Control of Indirect Cinematic Lighting. *Computer Graphics Forum* 27 (4), pages 1217–1223. http://dx.doi.org/10.1111/j.1467-8659.2008.01260.x. ↑ 17

[Obe10] J. Obert, F. Pellacini, and S. Pattanaik, 2010. Visibility Editing for All-Frequency Shadow Design. *Computer Graphics Forum* 29 (4), pages 1441–1449. http://dx.doi.org/10.1111/j.1467-8659.2010.01741.x. ↑ 16, 17, 51

[Oce00] M. Ocelot, D. Brunner, and J.-F. Laguionie, 2000. *Princes et Princesses*. Les Armateurs, Gébéka Films. ↑ 2

[Oce11] M. Ocelot, C. Rossignon, and P. Boëffard, 2011. *Les Contes de la Nuit*. Nord-Ouest Films, StudioCanal. ↑ 2

[Oka07] M. Okabe, Y. Matsushita, L. Shen, and T. Igarashi, October 2007. Illumination Brush: Interactive Design of All-Frequency Lighting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 171–180. IEEE Computer Society, USA. http://dx.doi.org/10.1109/PG.2007.9. ↑ 10

[Orz08] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, August 2008. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. *ACM Trans. Graph.* 27 (3), pages 1–8. http://dx.doi.org/10.1145/1360612.1360691. ↑ 18

[Pac08] R. Pacanowski, X. Granier, C. Schlick, and P. Poulin, 2008. Sketch and Paint-based Interface for Highlight Modeling. In C. Alvarado and M.-P. Cani, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association. http://dx.doi.org/10.2312/SBM/SBM08/017-023. ↑ 11

[Pap11] M. Papas, W. Jarosz, W. Jakob, S. Rusinkiewicz, W. Matusik, and T. Weyrich, January 2011. Goal-based caustics. *Computer Graphics Forum* 30 (2), pages 503–511. http://dx.doi.org/10.1111/j.1467-8659.2011.01876.x. ↑ 15

[Pau00] M. Pauly, T. Kollig, and A. Keller, 2000. Metropolis Light Transport for Participating Media. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000*, pages 11–22. Springer Vienna, Vienna. http://dx.doi.org/10.1007/978-3-7091-6303-0_2. ↑ 66

[Pel02] F. Pellacini, P. Tole, and D. P. Greenberg, July 2002. A User Interface for Interactive Cinematic Shadow Design. *ACM Trans. Graph.* 21 (3), pages 563–566. http://dx.doi.org/10.1145/566654.566617. ↑ 12, 13

[Pel07] F. Pellacini, F. Battaglia, R. K. Morley, and A. Finkelstein, June 2007. Lighting with Paint. *ACM Trans. Graph.* 26 (2). http://dx.doi.org/10.1145/1243980.1243983. ↑ 10

[Pel10] F. Pellacini, 2010. envyLight: An Interface for Editing Natural Illumination. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/1833349.1778771. ↑ 11

[Pha16] M. Pharr, W. Jakob, and G. Humphreys, 2016. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition. ↑ 8, 32, 75

[Phi19] J. Philip, M. Gharbi, T. Zhou, A. A. Efros, and G. Drettakis, July 2019. Multi-View Relighting Using a Geometry-Aware Network. *ACM Trans. Graph.* 38 (4). http://dx.doi.org/10.1145/3306346.3323013. ↑ 19, 82

[Phi21] J. Philip, S. Morgenthaler, M. Gharbi, and G. Drettakis, September 2021. Free-Viewpoint Indoor Neural Relighting from Multi-View Stereo. *ACM Trans. Graph.* 40 (5). http://dx.doi.org/10.1145/3469842. ↑ 19, 82

[Pla21] Playbae, 2021. *In My Shadow*. Alcon Interactive Group. ↑ 4

[Pou92] P. Poulin and A. Fournier, 1992. Lights from Highlights and Shadows. In *Proc. Symposium on Interactive 3D Graphics*, I3D '92, pages 31–38. ACM, New York, NY, USA. http://dx.doi.org/10.1145/147156.147160. ↑ 12

[Pou97] P. Poulin, K. Ratib, and M. Jacques, June 1997. Sketching Shadows and Highlights to Position Lights. In *Proc. Computer Graphics International*, pages 56–63. http://dx.doi.org/10.1109/CGI.1997.601272. ↑ 10

[Ram07] R. Ramamoorthi, D. Mahajan, and P. Belhumeur, January 2007. A First-Order Analysis of Lighting, Shading, and Shadows. *ACM Trans. Graph.* 26 (1), pages 2–23. http://dx.doi.org/10.1145/1189762.1189764. ↑ 15

[Rei26] L. Reinigier and C. Koch, 1926. *Die Geschichte des Prinzen Achmed*. ↑ 2

[Ric14] C. Richardt, J. Lopez-Moreno, A. Bousseau, M. Agrawala, and G. Drettakis, 2014. Vectorising Bitmaps into Semi-Transparent Gradient Layers. In *Proceedings of the 25th Eurographics Symposium on Rendering*, EGSR '14, pages 11–19. Eurographics Association, Goslar, DEU. http://dx.doi.org/10.1111/cgf.12408. ↑ 18

[Rit09] T. Ritschel, M. Okabe, T. Thormählen, and H.-P. Seidel, December 2009. Interactive Reflection Editing. *ACM Trans. Graph.* 28 (5), pages 129:1–129:7. http://dx.doi.org/10.1145/1618452.1618475. ↑ 13

[Rit10] T. Ritschel, T. Thormählen, C. Dachsbacher, J. Kautz, and H.-P. Seidel, July 2010. Interactive On-surface Signal Deformation. *ACM Trans. Graph.* 29 (4), pages 36:1–36:8. http://dx.doi.org/10.1145/1778765.1778773. ↑ 13

[RW26] H. J. Robert Wiene, Carl Mayer, 1926. *Das Cabinet des Dr. Caligari*. Descla-Bioscop. ↑ 2

[Sai90] T. Saito and T. Takahashi, September 1990. Comprehensible Rendering of 3-D Shapes. *SIGGRAPH Comput. Graph.* 24 (4), pages 197–206. http://dx.doi.org/10.1145/97880.97901. ↑ 33

[San18] P. E. Santos, R. Casati, and P. Cavanagh, 2018. Perception, Cognition and Reasoning about Shadows. *Spatial Cognition & Computation* 18 (2), pages 78–85. http://dx.doi.org/10.1080/13875868.2017.1377204. ↑ 5

[Sat05] M. Sattler, R. Sarlette, T. Mücken, and R. Klein, 2005. Exploitation of Human Shadow Perception for Fast Shadow Rendering. In *Proc. 2nd symposium on Applied perception in graphics and visualization*, APGV05, pages 131–134. Association for Computing Machinery. http://dx.doi.org/10.1145/1080402.1080426. ↑ 5

[Sch93] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg, 1993. Painting with Light. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 143–146. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/166117.166135. ↑ 10

[Sch04] G. Schwizgebel and M. Jean, 2004. *L'homme sans ombre*. GDS, Office national du film du Canada. ↑ 4

[Sch13] T.-W. Schmidt, J. Novák, J. Meng, A. S. Kaplanyan, T. Reiner, D. Nowrouzezahrai, and C. Dachsbacher, July 2013. Path-space Manipulation of Physically-based Light Transport. *ACM Trans. Graph.* 32 (4), pages 129:1–129:11. http://dx.doi.org/10.1145/2461912.2461980. ↑ 18

[Sch14] Y. Schwartzburg, R. Testuz, A. Tagliasacchi, and M. Pauly, July 2014. High-Contrast Computational Caustic Design. *ACM Trans. Graph.* 33 (4). http://dx.doi.org/10.1145/2601097.2601200. ↑ 15

[Sch16] T.-W. Schmidt, F. Pellacini, D. Nowrouzezahrai, W. Jarosz, and C. Dachsbacher, 2016. State of the Art in Artistic Editing of Appearance, Lighting and Material. *Computer Graphics Forum* 35 (1), pages 216–233. http://dx.doi.org/10.1111/cgf.12721. ↑ 9

[Sha01] R. Shacked and D. Lischinski, 2001. Automatic Lighting Design using a Perceptual Quality Metric. *Computer Graphics Forum* 20 (3), pages 215–227. http://dx.doi.org/10.1111/1467-8659.00514. ↑ 14

[Sin03] M. Singh and X. Huang, 2003. Computing Layered Surface Representations: An Algorithm for Detecting and Separating Transparent Overlays. In *Proceedings of Computer Vision and Pattern Recognition*, pages 11–18. http://dx.doi.org/10.1109/CVPR.2003.1211446. ↑ 18

[Slo02] P.-P. Sloan, J. Kautz, and J. Snyder, July 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph.* 21 (3), pages 527–536. http://dx.doi.org/10.1145/566654.566612. ↑ 10

[Sub17] T. Subileau, N. Mellado, D. Vanderhaeghe, and M. Paulin, February 2017. Ray-Portals: A Light Transport Editing Framework. *Vis. Comput.* 33 (2), pages 129–138. http://dx.doi.org/10.1007/s00371-015-1163-2. ↑ 17, 18, 60

[Tar17] Tarsier Studios, 2017. *Little Nightmares*. Bandai Namco Entertainment. ↑ 4

[vC14] A. von Chamisso, 1814. *Peter Schlemihls wundersame Geschichte*. ↑ 4

[Vea97] E. Veach, 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. thesis, Stanford University, Stanford, CA, USA. ↑ 27, 29, 30, 46, 56, 57, 58, 66

[Vic21] D. Vicini, S. Speierer, and W. Jakob, August 2021. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *ACM Trans. Graph.* 40 (4), pages 108:1–108:14. http://dx.doi.org/10.1145/3450626.3459804. ↑ 16

[Vol06] C. Volckman, R. Lener, A. Soumache, and A. Vonarb, 2006. *Renaissance*. On Animation Studios, Millimages, LuxAnimation, Pathé Distribution. ↑ 2, 3

[Wan92a] L. Wanger, 1992. The Effect of Shadow Quality on the Perception of Spatial Relationships in Computer Generated Imagery. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, I3D '92, pages 39–42. Association for Computing Machinery, New York, NY, USA. http://dx.doi.org/10.1145/147156.147161. ↑ 5

[Wan92b] L. R. Wanger, J. A. Ferwerda, and D. P. Greenberg, 1992. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Computer Graphics and Applications* 12 (3), pages 44–58. http://dx.doi.org/10.1109/38.135913. ↑ 5

[Wan04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, April 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13 (4), pages 600–612. http://dx.doi.org/10.1109/TIP.2003.819861. ↑ 53

[Wil78] L. Williams, August 1978. Casting Curved Shadows on Curved Surfaces. *SIGGRAPH Comput. Graph.* 12 (3), pages 270–274. http://dx.doi.org/10.1145/965139.807402. ↑ 19

[Zel21] T. Zeltner, S. Speierer, I. Georgiev, and W. Jakob, August 2021. Monte Carlo Estimators for Differential Light Transport. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40 (4). http://dx.doi.org/10.1145/3450626.3459807. ↑ 16

[Zha19] C. Zhang, L. Wu, C. Zheng, I. Gkioulekas, R. Ramamoorthi, and S. Zhao, November 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph.* 38 (6). http://dx.doi.org/10.1145/3355089.3356522. ↑ 15, 16

[Zha20] C. Zhang, B. Miller, K. Yan, I. Gkioulekas, and S. Zhao, July 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39 (4), pages 143:1–143:19. http://dx.doi.org/10.1145/3386569.3392383. ↑ 16

[Zha21] C. Zhang, Z. Yu, and S. Zhao, July 2021. Path-Space Differentiable Rendering of Participating Media. *ACM Trans. Graph.* 40 (4), pages 76:1–76:15. http://dx.doi.org/10.1145/3450626.3459782. ↑ 16