PATH MANIPULATION STRATEGIES FOR RENDERING DYNAMIC ENVIRONMENTS

Adina Andreea Bizdideanu

DOCTOR OF ENGINEERING

IN DIGITAL MEDIA

Centre for Digital Entertainment

Bournemouth University

October 2018

Abstract

The current work introduces path manipulation as a tool that extends bidirectional path tracing to reuse paths in the temporal domain. Defined as an apparatus of sampling and reuse strategies, path manipulation reconstructs the subpaths that compose the light transport paths and addresses the restriction of static geometry commonly associated with Monte Carlo light transport simulations. By reconstructing and reusing subpaths, the path manipulation algorithm obviates the regeneration of the entire path collection, reduces the computational load of the original algorithm and supports scene dynamism.

Bidirectional path tracing relies on local path sampling techniques to generate the paths of light in a synthetic environment. By using the information localized at path vertices, like the probability distribution, the sampling techniques construct paths progressively with distinct probability densities. Each probability density corresponds to a particular sampling technique, which accounts for specific illumination effects. Bidirectional path tracing uses multiple importance sampling to combine paths sampled with different techniques in low-variance estimators. The path sampling techniques and multiple importance sampling are the keys to the efficacy of bidirectional path tracing.

However, the sampling techniques gained little attention beyond the generation and evaluation of paths. Bidirectional path tracing was designed for static scenes and thus it discards the generated paths immediately after the evaluation of their contributions. Limiting the lifespan of paths to a generation-evaluation cycle imposes a static use of paths and of sampling techniques. The path manipulation algorithm harnesses the potential of the sampling techniques to supplant the static manipulation of paths with a generation-evaluation-reuse cycle. An intra-subpath connectivity strategy was devised to reconnect the segregated chains of the subpaths invalidated by the scene alterations. Successful intra-subpath connections generate subpaths in multiple pieces by reusing subpath chains from prior frames. Subpaths are reconstructed generically, regardless of the subpath or scene dynamism type and without the need for predefined animation paths. The result is the extension of bidirectional path tracing to the temporal domain.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Description ([unit of measurement])	Page
x	Point in 3D space	12
$\vec{\omega}$	Unit length direction	12
$\vec{N}_x$	Unit normal at point x	12
r	Distance/radius measured from some point x ($[m]$)	12
$\mathcal{S}$	Surface	12
dA	Area of an infinitesimal patch usually centred at some point x ($[m^2]$)	12
$\Omega(\vec{\omega})$	Solid angle ($[sr]$)	12
$\Omega^{\perp}(\vec{\omega})$	Projected solid angle ($[sr]$)	12
$\mathcal{H}^+(x)$	Upper hemisphere centred on x	12
$\mathcal{H}^-(x)$	Lower hemisphere centred on x	12
$d\vec{\omega}$	Differential solid angle ($[sr]$)	13
$\mathcal{E}_\lambda$	Energy of a photon with wavelength λ ($[J]$)	14
Q	Radiant energy ($[J]$)	14
Φ	Radiant power ($[W]$)	14
$E(x)$	Irradiance ($[W/m^2]$)	15
$B(x)$	Radiosity ($[W/m^2]$)	15
$I(\vec{\omega})$	Radiant intensity ($[W/sr]$)	15
$L(x, \vec{\omega})$	Radiance ($[W/(m^2 sr)]$)	15
θ	Polar angle measured from the normal at x	16
ϕ	Azimuth angle measured relative to the tangent at x	16
$d\vec{\omega}^{\perp}$	Projected differential solid angle ($[sr]$)	17
$f_s(x, \vec{\omega}_i, \vec{\omega}_o)$	Bidirectional scattering distribution function ($[sr^{-1}]$)	18
$f_r(x, \vec{\omega}_i, \vec{\omega}_o)$	Bidirectional reflectance distribution function ($[sr^{-1}]$)	18
$f_t(x, \vec{\omega}_i, \vec{\omega}_o)$	Bidirectional transmittance distribution function ($[sr^{-1}]$)	18
$\vec{T}_x$	Unit tangent at point x	19
$\vec{B}_x$	Unit bitangent at point x	20
$\vec{H}$	Unit halfway vector at point x	21
$\rho(x)$	Reflectance of a surface at point x	22
$\rho_{bh}(x)$	Bi-hemispherical reflectance of an isotropically illuminated surface	23
$\rho(x, \vec{\omega}_i)$	Directional-hemispherical reflectance for incident direction $\vec{\omega}_i$	23
$\tau(x, \vec{\omega}_i)$	Tracing function determining the first intersection from x along $\vec{\omega}_i$	24
$\alpha_{min}(x, \vec{\omega}_i)$	Boundary distance function associated with the tracing function	24
$\mathcal{M}$	Set of scene surfaces	24
$\mathcal{D}$	Set of unit directions	25
$\mathcal{R}$	Ray space	25
K	Scattering operator	26
$\mathcal{P}$	Propagation/geometric operator	26
T	Light transport operator	26
$\mathcal{I}$	Identity operator	27
$V(x \leftrightarrow x')$	Visibility function between points x and x'	28
$G(x \leftrightarrow x')$	Geometric factor between points x and x' ($[sr/m^2]$)	29

Symbol	**Description ([unit of measurement])**	**Page**
$\mathcal{F}_{i\to j}$	Form factor computing radiosity from patch i to patch j	30
$\mathcal{I}_j$	Measurement associated with pixel j	30
$W(x,\vec{\omega})$	Importance ($[W^{-1}]$)	30
$f_s^*(x,\vec{\omega}_i,\vec{\omega}_o)$	Adjoint bidirectional scattering distribution function ($[sr^{-1}]$)	31
K^*	Adjoint scattering operator	31
T_W	Importance transport operator	31
Ӽ	Set of paths of all finite lengths	32
$d\mu(\bar{x})$	Differential area-product measure defined on a path $\bar{x}$	32
$f_j(\bar{x})$	Measurement contribution function defined on a path $\bar{x}$	32
$d\mu(\mathcal{r})$	Differential throughput of a small cone of rays around ray $\mathcal{r}$ ($[m^2sr]$)	34
$\mu(\mathcal{D}_{\mathcal{r}})$	Throughput of a non-infinitesimal cone of rays $\mathcal{D}_{\mathcal{r}}$ ($[m^2sr]$)	34
$\mu_m^j(\mathcal{D}_{\bar{x}})$	Measurement contribution throughput on the set of paths $\mathcal{D}_{\bar{x}}$	34
$\mu_p(\mathcal{D}_{\bar{x}})$	Power throughput on the set of paths $\mathcal{D}_{\bar{x}}$ ($[W]$)	34
$\mu_s(\mathcal{D}_{\bar{x}})$	Scattering throughput on the set of paths $\mathcal{D}_{\bar{x}}$ ($[m^2sr]$)	34
$\mu_g(\mathcal{D}_{\bar{x}})$	Geometric throughput on the set of paths $\mathcal{D}_{\bar{x}}$ ($[m^2sr]$)	35
$\Gamma(x,\vec{\omega}_i,\vec{\omega}_o)$	Transport function used to transfer radiance ($[sr^{-1}]$)	39
$p(\bar{x})$	Probability density associated with sampling path $\bar{x}$	44
$\mathcal{E}[F_N]$	Expected value of estimator F_N	44
$\mathcal{V}[F_N]$	Variance of estimator F_N	44
$\beta[F_N]$	Bias of estimator F_N	44
$MSE[F_N]$	Mean squared error of estimator F_N	45
$\sigma[F_N]$	Standard deviation of estimator F_N	45
$\epsilon[F_N]$	Efficiency of estimator F_N	45
$\mathcal{t}[F_N]$	Time required to evaluate estimator F_N	45
$p(x_i)$	Surface area probability density used to sample vertex x_i	46
$p(\vec{\omega}_o)$	Solid angle probability density used to sample direction $\vec{\omega}_o$	46
$p^{\perp}(\vec{\omega}_o)$	Projected solid angle probability density used to sample $\vec{\omega}_o$	46
$p(x\to y)$	Transition probability density of going from state x to state y	47
ξ	Random number	48
q_i	Continuation probability of a path to be extended past vertex x_i	48
y_{s-1}	Connecting vertex of a light subpath	51
z_{t-1}	Connecting vertex of an eye subpath	51
$\bar{x}_{s,t}$	Light transport path sampled with s light vertices and t eye vertices	51
$w_{s,t}(\bar{x}_{s,t})$	Multiple importance sampling weighting function for path $\bar{x}_{s,t}$	52
$f(\bar{x})$	Image contribution function defined on a path $\bar{x}$	54
$\mathcal{h}_j(\bar{x})$	Filter function correlating the contribution of a path $\bar{x}$ with pixel j	54
$\mathcal{a}(\bar{x}\to\bar{y})$	Acceptance probability of a path $\bar{y}$ mutated from a path $\bar{x}$	55
$\mathcal{K}_r(\lVert x_s^*-x_s\rVert)$	Density estimation kernel of radius r	64
$\bar{\rho}$	Average scene reflectance	67
$C_{s,t}$	Contribution of a path $\bar{x}_{s,t}$	101
η_s	Throughput of a light subpath $y_0 \ldots y_{s-1}$	101
η_t	Throughput of an eye subpath $z_0 \ldots z_{t-1}$	101
$c_{s,t}$	Factors of the measurement contribution function reliant on $y_{s-1}z_{t-1}$	101
$C_{s,t}^*$	Unweighted contribution of a path $\bar{x}_{s,t}$	102
ϐ	Exponent of the power heuristic used to compute $w_{s,t}(\bar{x}_{s,t})$	106

Symbol	**Description ([unit of measurement])**	**Page**
φ_i	The i^{th} rendering frame	107
$\bar{x}_{s,t}^{\varphi_i}$	Path used in the rendering of frame φ_i	107
a_{i+1}	Anchor from which an invalid subpath is reconstructed	111
$x_0 \dots a_{i+1}$	First chain of an invalid subpath	117
$x_{i+2} \dots x_{n-1}$	Second chain of an invalid subpath	117
$\Lambda_{i+1,\mathscr{j}}$	Contribution of the tentative connection between a_{i+1} and $x_{i+2 \le \mathscr{j} \le n-1}$	118
$q_{i+1,\mathscr{j}}^{\Lambda}$	Probability with which $\Lambda_{i+1,\mathscr{j}}$ passes the sufficient throughput test	118
$\varkappa$	Threshold against which $\Lambda_{i+1,\mathscr{j}}$ is tested	118
$q_{i+1,\mathscr{j}}^{P}$	Probability for the tentative connection to occur	120
$q_{i+1,\mathscr{j}}$	Probability with which the intra-subpath connection is established	121
$\varsigma_{\mathscr{r}}$	The cost of sampling a ray $\mathscr{r}$	132
ς_{τ}	The cost of computing the first scene intersection by tracing a ray	132
ς_{η}	The cost of evaluating the throughput of a vertex	132
ς_{eval}	The cost of evaluating the contribution of a path $\bar{x}_{s,t}$	132
ς_{recon}	The reconstruction cost per subpath	133
$\varsigma_{q_{i+1,\mathscr{j}}^{\Lambda}}$	The cost of computing the probability $q_{i+1,\mathscr{j}}^{\Lambda}$	134
$\delta_{i+1,\mathscr{j}}$	Kronecker delta defined for the intra-subpath connection $a_{i+1} - x_{\mathscr{j}}$	134
$\delta_{v,\tau}$	Kronecker delta defined for the subminimal length of a subpath	135
ς_{SP}	Memory cost per average length subpath	172
ς_{φ_i}	Memory cost for all the subpaths used to render a frame	172

Acknowledgments

My fascination for light and illumination started with observing the change of colours in my mother’s irises, an effect which I can now explain through Rayleigh scattering and the concentration of melanin within the stroma. The first contact with the field of illumination occurred during my studies at PERCRO, a laboratory of Scuola Superiore Sant’Anna, when I came across articles like “Display of the earth taking into account atmospheric scattering” (Nishita et al. 1993). The idea of working on ray tracing was first suggested by Assistant Professor Franco Tecchia, whom I would like to thank for his enthusiasm and great support. Since then my quest has been to learn more about light and to develop a thorough understanding of the Monte Carlo light transport algorithms.

The opportunity to do so came with the offer made by the Centre for Digital Entertainment, The Media School, Bournemouth University, to study for the degree of Engineering Doctorate. The generous grant from the Engineering and Physical Sciences Research Council supported both my studies and the deepening of my knowledge about cultures. It is truly extraordinary to partake, as a student, in events such as world-renowned conferences and attend expert discussions around the globe. Consequently, I would like to thank my university supervisor Ian Stephenson for his trust vote during the interviews that followed my application for the doctoral programme. I would also like to thank my university supervisor Oleg Fryazinov for his consideration, trust and encouraging words. I thank both my university supervisors for their feedback on this work. I am thankful to the valuable and reliable professionals who took care of and abstracted the administrative aspects of my academic activities.

The Cornell box model, used to carry the first three tests examined in chapter 5, was constructed after the data provided by the Cornell University (2017). The 3D models utilised for the other two investigated tests were downloaded from CGTrader (2017).

The Engineering Doctorate framework required an industrial context in which to conduct the research activities. For me the industrial context was Optis, a software development company with considerable experience in the field of optical simulation. I discovered Optis at SIGGRAPH 2013 Exhibition and it represented an extraordinary opportunity to shape and conduct my research. I would like to thank Director Chris Grieve for his exceptionally broad perspective on things, which made possible my collaboration with Optis. I would also like to thank my industrial supervisor Nicolas Dalmasso, for his optimism and trust, which have conferred me great freedom in conducting my research. I also thank him for establishing connections with the French side research and development engineers. I thank Engineer Eric Humbert for the fruitful discussions on bidirectional path tracing and Markov chains. I thank Jacques Diringer for his kindness and assistance with the real-time technology. I would like to thank Dominique Chabaud for his kindness and help in discovering beautiful Provence. I thank Lorraine Qadeer for helping me settle in and for the numerous sweet delights.

I would like to thank Lecturer Professor Rodica Medan for her refusal to compromise, dedication, joyfulness and refinement in teaching English. I thank her and Lecturer Professor Vasile Prejmerean for their support in applying for the doctoral programme.

I thank my perspicacious mother and my creative father for their unexhausted love and support. I am amazed to see a “perpetuum mobile” type of dedication. I also thank my friends and relatives who supported me in countless ways throughout this study period.

I would like to extend my deepest gratitude to Adonai, for guiding me towards the Engineering Doctorate studies and for depositing unfailing trust in me. I am grateful to experience the fact that in Light there is no variableness, neither shadow of turning.

Chapter 1

Introduction

The current work introduces path manipulation as a tool that extends bidirectional path tracing to the temporal domain and thus supports the generation of a wide variety of illumination effects for input models subjected to conditions of dynamic geometry. Defined as an apparatus of sampling and reuse strategies, path manipulation exploits full light transport paths from a temporal perspective, with the purpose of addressing the restriction of static geometry commonly associated with Monte Carlo simulations.

Monte Carlo ray tracing is a ubiquitous class of global illumination algorithms that generates physically correct light transport solutions for a variety of complex lighting, scattering and geometric models. The pivotal feature of these algorithms is robustness, defined as input generality, physical plausibility and visual appeal. Though robustness motivates the employment of Monte Carlo ray tracing well beyond computer graphics, in numerous industries, it is however the effect of the underlying sampling techniques.

A Monte Carlo algorithm forms the image of a 3D model by randomly sampling light transport paths and computing Monte Carlo estimators (chapter 2, subsection 2.5.4). The sampling techniques (chapter 3, subsection 3.1.1) generate the light transport paths progressively, based on the information of the last sampled vertices, such as the local probability distribution. They also underlie the evaluation of the path contributions (chapter 3, subsection 3.1.2). However, the sampling techniques received little attention outside the generation of paths and the evaluation of their contributions. Monte Carlo ray tracing algorithms were designed for static environments and thus they discard the generated paths immediately after the evaluation of their contributions. Limiting the lifespan of paths to a generation-evaluation cycle imposes a static use of paths and of sampling techniques. Yet, the potential of the latter can be exploited beyond the path generation-evaluation cycle, through solutions that manipulate paths in a temporal aware manner. A Monte Carlo algorithm that processes light transport paths using both the spatial and the temporal coordinate avoids the regeneration of the entire path collection, reduces the execution costs and obviates the assumption of static geometry.

The balance between accuracy and time complexity has always been the driving force of both offline and real-time global illumination, though to different effect. In the case of offline global illumination, the efforts were directed towards attaining high degrees of accuracy (Whitted 1980; Cook et al. 1984; Kajiya 1986; Veach and Guibas 1994; Veach and Guibas 1997; Georgiev et al. 2012; Vorba et al. 2014; Šik et al. 2016). Having a strict millisecond frame budget (Bikker and van Schijndel 2013), real-time global illumination must attain high-speed executions (Damez et al. 2003), notwithstanding the cost of simplifying assumptions (Ritschel et al. 2012). For Monte Carlo ray tracing, the accuracy-performance balance gained even more significance with the engagement of the former in a multitude of industrial sectors like aerospace, automotive industry, electronics, light design, architecture, medicine, luxury goods and many others. The issue raised by the industry is no longer a trade-off, but rather a duality encompassing accuracy and contained execution time. Fast executions imply additional flexibility in accommodating dynamism, an appealing trait in most sectors as it permits the spatial altering of objects and thus an evolving perspective of the scene.

Monte Carlo ray tracing is computationally expensive. On the one hand, numerous sampling operations, visibility tests and contribution evaluations are required to produce results with an acceptable level of variance, i.e. without objectionable high frequency noise. On the other hand, Monte Carlo algorithms were designed to render static scenes and consequently they fail to accommodate dynamic scenes. The slightest changes of a scene are processed by fully recomputing the global illumination solution.

Research in offline Monte Carlo ray tracing produced a series of techniques aimed at reducing the variance of the light transport solution and with it the execution costs. Variance manifests itself as noise and it affects the image quality. It is the result of using insufficient paths to appropriately estimate the illumination of the environment.

Different Monte Carlo algorithms, including bidirectional path tracing, reuse paths across various measurements. Path mutations, probabilistic connections, combinatorial connections, additional sample extraction from existing paths are all techniques that reuse paths and consequently reduce the sampling costs and the volume of resources required by the global illumination computations. Besides path reuse, most algorithms employ techniques that either reduce the tracing cost per individual path (e.g. splitting and Russian roulette) or design estimators that reduce variance (e.g. control variates, importance sampling, stratified sampling, adaptive sampling or correlated estimators). Recent advancements use more sophisticated techniques, like image-space control variates (Rousselle et al. 2016) and other reconstruction strategies (Zwicker et al. 2015), to reduce variance. However, most of these algorithms operate on static environments or render certain effects without a true global illumination solution (Bagher et al. 2013).

Only recently, offline Monte Carlo ray tracing started to manifest an interest in exploiting temporal coherence and supporting animated sequences (Manzi et al. 2016).

A considerably larger volume of research has been conducted in classic ray tracing (chapter 2, subsection 2.6.2). The goal of a classic ray tracer is to compute the average radiance of each pixel, using only several primary and secondary rays. By harnessing the computational power of the stream processing technology and by adapting the creation and traversal of the acceleration structures to dynamic conditions, classic ray tracing can render dynamic scenes at interactive or even real-time frame rates. However, the existing algorithms only trace secondary bounces (e.g. shadow rays, specular/distributed reflections or transmissions) and exploit temporal coherence only by refitting the acceleration structure. Similarly, progressive Monte Carlo ray tracing uses the high throughput rates of the stream processing architectures to improve the accuracy-performance balance. Unlike classic ray tracing, the progressive algorithms compute full light transport solutions. Nevertheless, scene dynamism is not treated directly and can be expected solely as a consequence of performance improvements. In other words, temporal coherence is not a current objective for progressive methods.

The current work approaches the predominantly static nature of Monte Carlo light transport simulations by extending bidirectional path tracing towards reusing paths in the temporal domain. The tool used for this aim is a set of sampling and reuse strategies.

Sampling techniques and multiple importance sampling (Veach and Guibas 1995) are the keys to the robustness of bidirectional path tracing. The sampling techniques are used to generate and evaluate the paths of light in a synthetic environment. On the one hand, the punctiform nature of the sampling techniques abstracts bidirectional path tracing from the input complexity. On the other hand, each sampling technique accounts for a specific group of illumination effects. Multiple importance sampling optimally combines paths, sampled with various techniques, in low-variance estimators.

The main outcome of the current work is a path manipulation algorithm that uses both sampling techniques and multiple importance sampling to reconstruct and reuse paths across different frames. The goal is to inject scene dynamism in Monte Carlo light transport simulations and supplant the static manipulation of paths with a generation-evaluation-reuse cycle. Unlike most path reuse solutions (chapter 2, subsection 2.6.3), the proposed algorithm reconstructs and reuses paths independently of the subpath and scene dynamism types. That is, light and eye subpaths are processed generically and regardless of whether the transformations affected the camera, a light source or other scene objects. Moreover, a priori knowledge about the animation paths is not required.

Three major steps define the reconstruction process. Firstly, the subpaths invalidated by the transformations of the geometry are identified (chapter 3, section 3.3). The geometric transformations invalidate either a light subpath, an eye subpath or the connecting edge between the two. The disconnection between two subpaths is easily solved by reconnecting each subpath to other homologues. The invalid light and eye subpaths are of two types, namely in-scope and out-of-scope subpaths. An in-scope subpath, is a subpath with an edge obstructed by a dynamic object. An out-of-scope subpath, is a subpath invalidated by the movement of one of its vertices. All the invalid subpaths could be identified by searching for new intersections on the subpath edges. However, optimizations are applied on a per type basis (chapter 4, subsection 4.3.1).

Secondly, the invalid subpaths are reconstructed in a manner that restores the disrupted particle flow (chapter 3, section 3.4). A novel intra-subpath connectivity strategy is used to reconnect the segregated subpath chains into coherent subpaths (chapter 3, subsection 3.4.2). The subpath chains are sections of an invalid subpath, which the novel strategy processes independently of the homologue that is connected to the invalid subpath. The gist is to concurrently exclude and reuse standalone subpath chains that would bring no contribution unless reconnected to light or eye chains. Should an intra-subpath connection fail, the invalid subpath will be regenerated from the new intersection point determined on the disrupted subpath edge. The regeneration is performed through the classic, local path sampling techniques (Veach 1998, p. 226).

Lastly, the throughput of the reconstructed subpaths is computed and the contributions of the resultant light transport paths are evaluated (chapter 4, subsection 4.3.4). The collateral subpaths, i.e. the subpaths with intact geometric structures but invalid contributions, are reconnected and re-evaluated as well (chapter 4, subsection 4.3.5).

1.1 Industrial context

The current research was conducted within Optis, a software development company with considerable experience in optical simulation. Optis provides a variety of services that encompasses lighting visualisation simulation for early design validation, human vision simulation, optical shape design, digital vision system position modelling and optimisation, optical and photometric simulation within CAD/CAM tools, field of view simulation for vehicle design and optical regulation, light management via virtual reality and dynamic distortion visualisation for windshield design. The core technology which supports the light simulation products is a physically-based rendering engine.

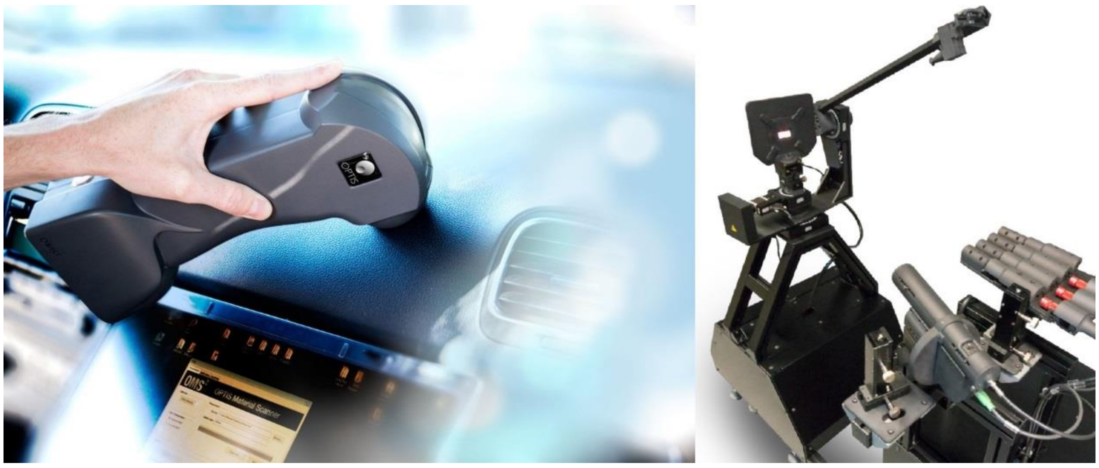


Figure 1.1*: The OMS2 (left) and OMS4 (right) material scanners (courtesy of Optis).*

The rendering engine can simulate a comprehensive range of optical effects for scene configurations inclusive of real-world illumination, true material properties and human eye dynamics. The illumination models can simulate light, both indoors and outdoors, with accurate levels of emission and spectrum distribution. Material properties can be either modelled analytically or measured physically with the in-house bidirectional scattering distribution function scanners (OMS2/OMS4). Materials are defined by spectra and can exhibit volumetric and heterogeneous properties. The OMS2 scanner is portable and acquires the reflectance of opaque objects with more than 1 million bytes. The OMS4 scanner comprises a fixed installation and measures reflectance, transmission and subsurface scattering with a maximum dynamic range of 10^8 bytes. The supported sensors can have either a pure radiometric response or a photometric one, i.e. a response which includes the sensitivity of a human eye model, with idiosyncrasies like age and visual deficiencies. The photometric sensors can be stimulated by any wave in the electromagnetic spectrum, whereas the radiometric ones are limited to the visible range. The human eye response is especially important for the industries that carry numerous visual acuity, perception and safety tests throughout the design process. Figure 1.1 illustrates the OMS2 and OMS4 material scanners.

The complex scene configuration is provided as input to the physically-based renderer, which uses unidirectional Monte Carlo ray tracing and/or photon mapping to generate the light transport solution. Though robust, the results entail protracted executions. Figure 1.2 depicting the 1800×1600 image of a centre high-mount stop lamp rendered in 16h:32m:25s, using 9 samples per pixel and 3 workstations with 16 threads launched.

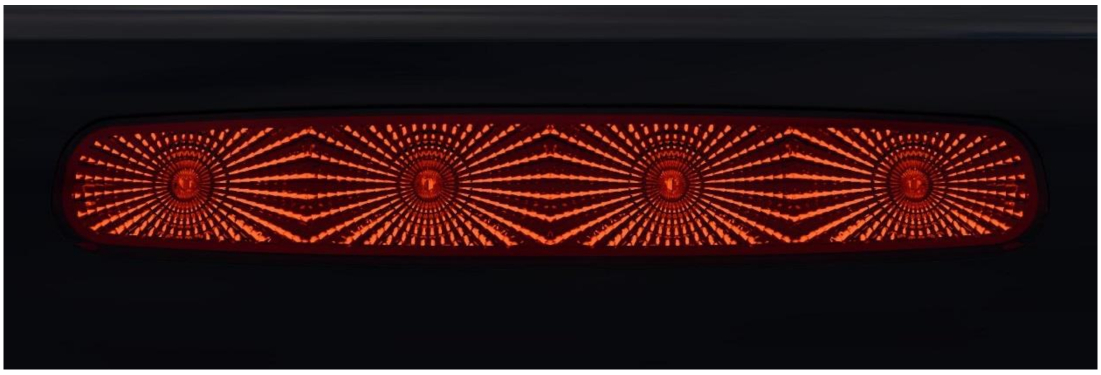


Figure 1.2*: Light propagation in a centre, high-mount stop lamp (courtesy of Optis).*

For its interactive and real-time products, Optis relies on a rendering engine that computes both direct and global illumination solutions. The direct illumination is computed in real-time with forward or deferred rendering algorithms. The global illumination is computed either with a real-time recursive ray tracer or with a progressive path tracer. Before the current research, the latter algorithms were the only bridging solutions between the offline and the real-time area of development. Recursive ray tracing and path tracing are explained in chapter 2 (subsections 2.5.3 & 2.5.4.1).

From Optis' perspective, the main challenge was to reduce the execution time of the offline renderer, while improving the accuracy of the real-time renderer. Accuracy, input generality, variety of illumination effects and scene dynamism were the principal factors that shaped the current research. The offline renderer accommodated arbitrary levels of scene complexity and generated unbiased results, whose errors could be computed and used in analysis stages, like industrial standard conformity evaluation and chromaticity studies. Accuracy, input generality and variety of illumination effects exacted high computational costs. Being computationally less prohibitive, the real-time renderer could support dynamism, at the cost of a limited range of illumination effects. The current work was developed as a solution that addresses all these conditions and injects novelty and progress in the development process. The target was to meet market demands with a product that supports dynamism with the quality of an offline renderer.

1.2 Summary of contributions

The principal contributions of the current work can be summarized as indicated below:

- The path manipulation algorithm extends bidirectional path tracing to reuse paths in the temporal domain and thus explicitly supports the generation of a wide variety of illumination effects for generic models subjected to dynamism.
- The current work introduces the concepts of immutable contribution and path validity as the norms used to identify the subpaths which can be directly reused in the rendering of the altered scene. An immutable contribution implies a valid path, whose lifespan can be extended from a generation-evaluation cycle to a generation-evaluation-reuse one. The subpaths that breach the validity criteria are identified as invalid and the notion of anchor is used in their reconstruction.
- The path manipulation algorithm implements an intra-subpath connectivity strategy which reconnects the dysfunctional chains of the invalid subpaths in a manner that maximizes path reuse. The novel strategy uses the scattering and the stochastic properties of the connecting vertices to establish intra-subpath connections. Unlike conventional path generation methods, which use Russian roulette only to control the subpath lengths, the proposed strategy applies a two-dimensional rejection test also during the connection process. Effectively, the intra-subpath connectivity strategy generates subpaths in multiple pieces.
- The path manipulation algorithm reconstructs and reuses paths independently of the subpath or scene dynamism type and without the need for predefined animation paths. The light and eye subpaths are processed generically, without regard to whether the camera, a light source or another object was transformed. These capabilities benefit especially applications that require direct interaction with the 3D environment, like the light simulation tools for CAD/CAM systems.
- The light transport framework provided Optis with the first implementation of bidirectional path tracing. It also included the first, in-house implementation of the recursive multiple importance sampling schema (van Antwerpen 2011b). The path generation approach proposed by Novák et al. (2010) was adapted to sample the light transport paths. Unlike the state-of-the-art method, the version developed in the light transport framework terminates subpaths stochastically.

- The light transport framework was designed modularly and thus the path manipulation algorithm ports bidirectional path tracing to the temporal domain by replacing only the path generation phase of its pipeline. In reconstructing rather than generating paths, the path manipulation algorithm supports scene dynamism with the quality of an offline renderer. This capability extends the applicability of the path manipulation algorithm to many sectors, among which aerospace, automotive industry, light design, architecture and cultural heritage.

1.3 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 analyses the problem domain of the current work and the decisions that shaped the path manipulation algorithm. The discussion starts by defining the solid angle and the radiometric quantities. Surface scattering peruses the bidirectional scattering distribution function, as the main tool that characterizes the local behaviour of light at encountering objects. The local illumination mechanism is used as a basis for the inspection of the rendering equation and its alternative formulations. The presented light transport theory supports the discrimination between miscellaneous global illumination algorithms and the self-contained theoretical formulation of the path manipulation algorithm. The review of various classes of global illumination algorithms reflects the technical review carried for Optis and the decisional process that led to the selection of bidirectional path tracing as the development foundation of the current work. The chapter concludes by inspecting the current work relative to the recent advances in light transport simulation.

Chapter 3 discusses the theoretical foundation of the path manipulation algorithm. The local path sampling techniques and the sample contribution evaluation schema are detailed based on the original formulation proposed by Veach (1998, p. 302-307). The concepts of immutable contribution and path validity are defined using the standard framework in which the path sampling techniques operate. These two concepts identify the paths that can be immediately reused in the rendering of the transformed scene. The discussion proceeds by exploring the first step of the path manipulation algorithm, namely the identification of the invalid subpaths and the computation of the anchors. The reconstruction of the invalid subpaths is structured in three scenarios, based on whether the anchor follows the first vertex (primary anchor reconstruction), replaces the last vertex (terminus anchor reconstruction) or divides a subpath into two chains (two-chain reconstruction). The latter scenario presents the intra-subpath connectivity strategy and each scenario discusses the evaluation of the reconstructed subpaths. The chapter closes with a high-level description and an analysis of the proposed algorithm.

Chapter 4 details the implementation of the light transport framework. The discussion starts with an overview of the implemented Monte Carlo ray tracing algorithms. The light transport framework was designed modularly and thus the focus of this chapter is on the building blocks that define the pipeline of both bidirectional path tracing and the path manipulation algorithm. The description of the standard Monte Carlo pipeline subsumes not just the bidirectional path tracer, but all the developed Monte Carlo ray tracing algorithms. The path generation, contribution evaluation and image synthesis phases of the standard pipeline are analysed in terms of building block descriptions. Having examined most of the building blocks, the rest of the chapter delineates the block that supplants the path generation phase and defines the path manipulation algorithm. The chapter concludes with a schematic flow of the propounded algorithm.

Chapter 5 examines the results of the path manipulation algorithm. Several tests on the Cornell box are used to benchmark the proposed algorithm against bidirectional path tracing. These tests analyse the performance gap between the baseline and the novel algorithm and report on the information gathered from the reconstruction process. The carried inspection also discusses the factors that influence the intra-subpath connectivity. The subsequent tests were ideated to reflect two of the Optis projects and emphasize the capabilities of the path manipulation algorithm. One test was devised for product assembling/disassembling sequences in CAD/CAM systems and it subjects several body parts to affine transformations. The other test combines object, light source and camera transformations in a building interior redesign scenario. Both tests analyse the performance and the reconstruction information associated with the path manipulation algorithm. Their objective is to stress the ability of the novel algorithm to reconstruct paths generically and without the need for predefined animation paths.

Chapter 6 reflects on the limitations of the path manipulation algorithm and explores future developments. Chapter 7 concludes with an outline of the original contributions.

Chapter 2

Background and literature review

This chapter analyses the problem domain of the current work and expounds the design decisions that shaped the development of the path manipulation algorithm. It evinces the limitations of various light transport algorithms and reflects the decisional process that led to selecting bidirectional path tracing as the development foundation of the path manipulation algorithm. Ideated as an apparatus of sampling and reuse strategies, path manipulation extends the use of light transport paths to the temporal domain, with the purpose of injecting scene dynamism in Monte Carlo light transport simulations.

Contextual analysis starts by defining the solid angle and the radiometric quantities. Surface scattering builds on the radiometric notions and introduces the bidirectional scattering distribution function, as the main tool that defines the local behaviour of light at encountering objects. Surface reflectance is further characterized through the definition of several derived quantities. The rendering equation is derived from the scattering equation and its alternative formulations are presented in the light transport section. The surface domain and the path integral formulations detail the structure of the rendering equation and underlie different approaches to interpreting and estimating it. The light transport theory (sections 2.1 - 2.4) serves to discriminate between global illumination algorithms and to theorize the path manipulation algorithm (chapter 3). Section 2.5 discusses various classes of global illumination algorithms and reflects the technical review, carried for Optis, that led to selecting bidirectional path tracing as the development foundation of the current work. The chapter closes by inspecting the current work relative to the recent advances in light transport simulation (section 2.6).

2.1 Solid angle

The solid angle is a central notion, used either as a parameter in the definition of certain radiometric quantities or as a measure in the specification of light transport operations.

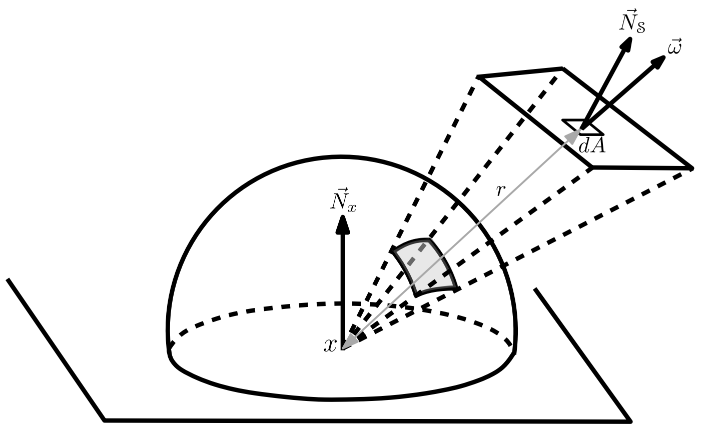


***Figure 2.1**: Measured in steradians $[sr]$, the solid angle expresses the size of a surface $\mathcal{S}$ as it is perceived from an observation point x, found at a distance r from the surface.*

The *solid angle* subtended by a surface $\mathcal{S}$ at a given point x is determined by projecting $\mathcal{S}$ onto a unit sphere centred around x and integrating the resulting cone of directions with respect to the surface area. That is, the solid angle is defined by the ratio between the area $\mathcal{S}$ covers on a unit sphere and the squared value of the distance r from x to $\mathcal{S}$:

$$\Omega(\vec{\omega}) = \iint_{\mathcal{S}} \frac{\left|\vec{\omega} \cdot \vec{N}_{\mathcal{S}}\right| dA}{r^2} \tag{2.1}$$

where $\vec{\omega}$ is a unit length direction emanating from x and $\vec{N}_{\mathcal{S}}$ is the unit normal of $\mathcal{S}$ at an infinitesimal patch of area dA. Figure 2.1 illustrates the ideas behind the solid angle.

The *projected solid angle* (Veach 1998, p. 77) is determined by further projecting the set of directions onto the space tangent at x and finding the area of the resulting surface. The tangent space is given by all the vectors that are orthogonal on the surface normal at point x. For a specific direction $\vec{\omega}$ the projected solid angle is expressed as:

$$\Omega^{\perp}(\vec{\omega}) = \left|\vec{N}_x \cdot \vec{\omega}\right| \Omega(\vec{\omega}) \tag{2.2}$$

By definition, the solid angle assimilates all the directions that connect point x to surface $\mathcal{S}$. The tangent space divides the sphere of directions that emanate from x into an upper hemisphere $\mathcal{H}^{+}(x)$ and a lower hemisphere $\mathcal{H}^{-}(x)$. The upper hemisphere subsumes the unit vectors situated on the same side as the surface normal at x. The lower hemisphere contains the unit vectors situated on the oppositely directed side of the normal. The projected solid angle, for any of the two hemispheres, is found by projecting $\mathcal{H}^{+/-}(x)$ onto the tangent space and computing the area of the resulting disk.

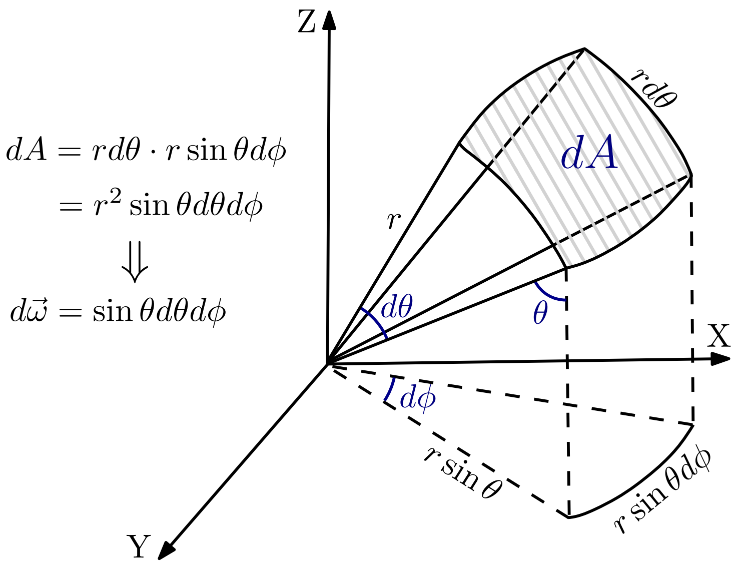

Z
X
Y
$dA = rd\theta \cdot r\sin\theta d\phi$
$= r^2 \sin\theta d\theta d\phi$
$\Downarrow$
$d\vec{\omega} = \sin\theta d\theta d\phi$
$r d\theta$
dA
r
$d\theta$
θ
$d\phi$
$r \sin\theta$
$r \sin\theta d\phi$

***Figure 2.2**: The derivation of the infinitesimal solid angle using spherical coordinates.*

The projected solid angle can also be derived for arbitrary distributions of directions. In the context of Monte Carlo integration, the projected solid angle may be used as a measure for the probability densities from which extension rays are sampled. Generally, the solid angle is used as a parameter in the definitions of the radiometric quantities.

The *differential solid angle* is another useful concept, usually expressed in the spherical coordinate system, which is the prevalent surface framework in light transport theory:

$$d\vec{\omega} = sin\theta d\theta d\phi \tag{2.3}$$

The rationale of the notation $d\vec{\omega}$ is that the differential solid angle encloses a very small cone of directions, which can be simply characterized by the given direction $\vec{\omega}$. Conversion (2.3) is particularly useful for sampling distribution functions, like the bidirectional scattering distribution function or the emission distribution of a light source. Pharr and Humphreys (2004), use extensively equation (2.3) to analytically draw samples from probability distributions. Figure 2.2 derives the differential solid angle by defining the area of an infinitesimal patch in the spherical coordinate system.

2.2 Radiometric quantities

Radiometry is the science of measuring electromagnetic radiation, of correlating measured results with reference data and of validating theoretical hypotheses. The radiometric quantities are descriptors of light behaviour. Each quantity defines the distribution of energy based on a set of parameters, which is indicative of a specific light phenomenon. The current section discusses the prominent radiometric quantities.

2.2.1 Radiant power

The elementary particle in lighting is the photon. In vacuum, a photon moves with the speed of light and has an energy that varies inversely proportional to its wavelength λ:

$$\mathcal{E}_\lambda = \frac{hc}{\lambda} \tag{2.4}$$

where h refers to Planck's constant and c represents the speed of light in empty space.

The *radiant energy* (Joules $[J]$) is the energy of the total number of photons emitted in the surrounding environment by a light source. It is expressed as (Jensen 2001, p. 13):

$$Q = \int_{\mathcal{D}_e} N_\lambda \mathcal{E}_\lambda d\lambda \tag{2.5}$$

where $\mathcal{D}_e$ is the emission region and N_λ is the number of photons that have wavelength λ and energy $\mathcal{E}_\lambda$. The radiant energy is a function of time and should have also been integrated over some time interval $[t_i, t_{i+1}]$. However, only systems in equilibrium are considered and thus the time dimension can be omitted. A system in equilibrium entails a constant density of photons, which does not change with time (Veach 1998, p. 81).

The *radiant power* (Watts $[J/s]$) is defined as the flow of radiant energy per unit time:

$$\Phi = \frac{dQ}{dt} \tag{2.6}$$

2.2.2 Irradiance, radiant exitance and radiant intensity

Irradiance ($[W/m^2]$) generally measures incident radiation (Veach 1998, p. 81). It is a function of position and denotes the radiant power that illuminates a unit surface area:

$$E(x) = \frac{d\Phi}{dA(x)} \tag{2.7}$$

As a positional measure, it quantifies the incident radiation relative to the normal defined at the observation point. Conversely, the quantity that measures the radiant power exiting a unit surface area is referred to as *radiant exitance* (M) or *radiosity* (B).

Radiant intensity ($[W/sr]$) is a directional measure that quantifies incident or exitant radiation. It is the power received or emitted per unit solid angle (Jensen 2001, p. 14):

$$I(\vec{\omega}) = \frac{\mathrm{d}\Phi}{d\vec{\omega}} \tag{2.8}$$

When it quantifies exitant radiation, the radiant intensity refers to phenomena like emission, reflection or transmission. When a surface is lit by a cone of directions, which encloses a unit solid angle at the apex, the radiant intensity measures incident radiation.

2.2.3 Radiance

Radiance (Veach 1998, p. 81) is the cardinal radiometric quantity of light transport theory. It quantifies the amount of light emanated from the cone of directions subsumed in a unit solid angle that reaches a virtual unit area surface, which is perpendicular on the emission axis. Usually, the orientation of the emitting surface is not coincident with the emission axis and the unit area around the illuminating point is projected onto the emission axis. Radiance is the radiant power per unit projected area and unit solid angle:

$$L(x, \vec{\omega}) = \frac{d^2\Phi}{dA(x)\left|\vec{N}_x \cdot \vec{\omega}\right| d\vec{\omega}} \tag{2.9}$$

where $\left|\vec{N}_x \cdot \vec{\omega}\right|$ is the cosine between the normal at x and the emission axis $\vec{\omega}$. Figure 2.3 depicts the geometry involved in the scattering of radiance between two surfaces.

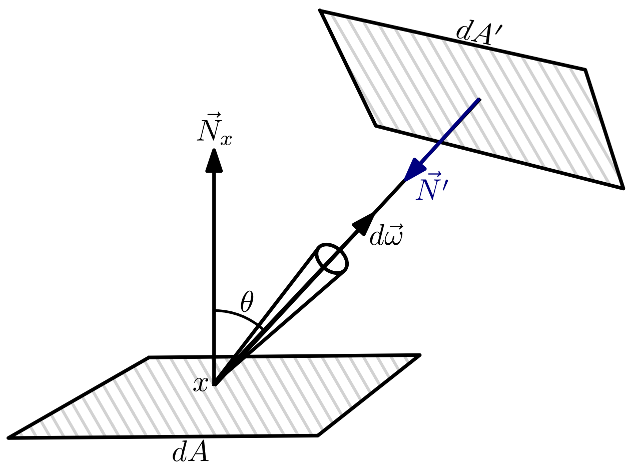


Figure 2.3: *The radiance that arrives at dA' from dA along the emitting direction $d\vec{\omega}$.*

Radiance ($[W/(m^2 sr)]$) is a five-dimensional quantity, evaluated at a given position and for a given direction. The directional parameter can be specified in spherical coordinates, as a pair of angles (θ, ϕ). The polar angle θ is measured from the normal at x, whereas the azimuth angle ϕ is measured relative to the tangent at x. In vacuum, radiance is constant along a propagation direction (Jensen 2001, p. 15). The latter property is extremely important, since all the ray tracing algorithms apply it within the environments that lack participating media or have constant indices of refraction.

An alternative to projecting the unit area onto the plane defined by the emission axis, is to integrate $|\vec{N}_x \cdot \vec{\omega}|$ in the solid angle and use the projected solid angle as parameter. This interpretation may be more cogent as it uses the natural area (Veach 1998, p. 82).

Radiance is computed by both the local and the global illumination framework. The bidirectional scattering distribution function (subsection 2.3.1) relates radiance to irradiance. The scattering operator (subsection 2.4.1) computes the exitant radiance given the incident counterpart, whereas the propagation operator (subsection 2.4.1) performs the opposite transformation. The distinction between incident and exitant radiance (Veach 1998, p. 83) concerns the transformations undergone by the measured photons. At each surface, the directions and the wavelengths of the scattered photons depend on the surface properties. Hence, the two types of radiance regard the attributes manifested by the photons, at the observed time and location. The rendering equation (subsection 2.3.4) computes radiance recursively, at different locations in the scene.

2.3 Surface scattering

Surface appearance is an important visual cue in discriminating between the nature of different objects. The appearance of an object is controlled by its material character and by the finish of its surface. The material character is related to the atomic structure of an object and it generally determines the classification of the latter into dielectric, metal or composite. The finish of a surface refers to the microscopic geometry of the surface. Most real-world objects are not optically smooth, but have varying degrees of roughness. For instance, Rayleigh criterion states that a surface is optically smooth only if it has a roughness less than $\lambda/(8cos\theta)$, where λ is the wavelength of the incident light and θ is the incident/polar angle. Together, material character and surface finish, determine the behaviour of light at intersecting the surface of an object.

2.3.1 The bidirectional scattering distribution function

The *bidirectional scattering distribution function* (BSDF) is a mathematical construct that describes the local light-scattering properties of a surface. The local scattering process begins with an infinitesimal surface being illuminated by a cone of incident directions. Assume the observed surface is centred on a point x and the cone of incident directions describes a unit solid angle around a direction $\vec{\omega}_i$. The radiometric quantity which characterizes the light that arrives at x from $\vec{\omega}_i$ is the incident radiance:

$$L_i(x, \vec{\omega}_i) = \frac{d^2\Phi}{dA(x)\left|\vec{N}_x \cdot \vec{\omega}_i\right| d\vec{\omega}_i}$$

The irradiance measured at point x due to direction $\vec{\omega}_i$ can be expressed as follows:

$$dE(x, \vec{\omega}_i) = L_i(x, \vec{\omega}_i)\left|\vec{N}_x \cdot \vec{\omega}_i\right| d\vec{\omega}_i = L_i(x, \vec{\omega}_i) d\vec{\omega}_i^{\perp} \qquad (2.10)$$

The direction $\vec{\omega}_i$ implies that the light measured at x arrives from a given location in space. The proportionality between irradiance and projected differential solid angle confirms the idea that incident light is transformed based on the surface properties. The projected differential solid angle $d\vec{\omega}^{\perp}$ is related to the differential solid angle $d\vec{\omega}$ via:

$$d\vec{\omega}^{\perp} = \left|\vec{N}_x \cdot \vec{\omega}\right| d\vec{\omega} \qquad \forall\vec{\omega} \qquad (2.11)$$

From the impact point x, light is scattered along a given propagation direction $\vec{\omega}_o$. The quantity which describes the light leaving a unit area in direction $\vec{\omega}_o$, is the exitant radiance. The proportion which relates the exitant radiance to irradiance, according to the light-scattering properties of the surface, is exactly the BSDF (Veach 1998, p. 85):

$$f_s : \mathcal{H}^{+/-}(x) \times \mathcal{H}^{+/-}(x) \to \mathbb{R},\ f_s(x, \vec{\omega}_i, \vec{\omega}_o) = \frac{dL_o(x, \vec{\omega}_o)}{dE(x, \vec{\omega}_i)} = \frac{dL_o(x, \vec{\omega}_o)}{L_i(x, \vec{\omega}_i) d\vec{\omega}_i^{\perp}} \quad (2.12)$$

The BSDF is generalized to both the upper and the lower hemisphere, meaning that it can describe both the reflection and the transmission phenomenon. By restricting the BSDF to the upper or the lower hemisphere, the bidirectional reflectance distribution function, respectively the bidirectional transmittance distribution function is obtained.

2.3.2 The bidirectional reflectance distribution function

The *bidirectional reflectance distribution function* (BRDF) is obtained by restricting the domain of the BSDF to the incident and reflected hemispheres (Veach 1998, p. 86):

$$f_r : \mathcal{H}_i(x) \times \mathcal{H}_r(x) \to \mathbb{R}$$

The omission of the signed superscript refers to the fact that the BRDF can be defined on either the upper or the lower hemisphere. Reflectance in the upper hemisphere describes the most habitually assumed type of reflection, like that from opaque objects. On the other hand, reflectance in the lower hemisphere may entail scenarios like the total internal reflection on the back of a raindrop, during the formation of the rainbow.

The *bidirectional transmittance distribution function* (BTDF) is defined by limiting the domain of the BSDF to the incident and transmitted hemispheres (Veach 1998, p. 87):

$$f_t : \mathcal{H}_i(x) \times \mathcal{H}_t(x) \to \mathbb{R}$$

As opposed to the incident and reflected hemispheres, which are located on the same side of the surface, the incident and transmitted hemispheres are located on opposite sides of the surface. Nevertheless, both $\mathcal{H}_i(x)$ and $\mathcal{H}_t(x)$ can refer to either $\mathcal{H}^+$ or $\mathcal{H}^-$.

The BRDF possesses properties that do not always hold for the BTDF or the BSDF.

2.3.2.1 Properties of the BRDF

An important property of the BRDF, known as the *Helmholtz law of reciprocity*, states that the BRDF is independent of the direction in which light flows, i.e. it is symmetric:

$$f_r(x, \vec{\omega}_i, \vec{\omega}_o) = f_r(x, \vec{\omega}_o, \vec{\omega}_i) \tag{2.13}$$

This property is fundamental to numerous global illumination algorithms, because it enables the generation of transport paths from both the light sources and the camera.

Another property of the BRDF refers to the *conservation of energy*. A surface cannot reflect more light than it receives, i.e. exitant energy must not exceed incident energy:

$$\int_{\mathcal{H}_r} f_r(x, \vec{\omega}_i, \vec{\omega}_o) \left|\vec{N}_x \cdot \vec{\omega}_o\right| d\vec{\omega}_o \leq 1 \qquad \forall \vec{\omega}_i \in \mathcal{H}_i \tag{2.14}$$

BRDFs characterized by these properties are physical and thus describe real surfaces. Veach (1998, p. 175) generalizes these two principles to surfaces with arbitrary BSDF.

2.3.2.2 Parameterizations of the BRDF

Definition (2.12) expresses the BSDF in terms of the unit directions $\vec{\omega}_i$ and $\vec{\omega}_o$. Hence, both the BRDF and the BTDF can be specified using this *unit direction representation*.

Another widely used parameterization, adopts the spherical coordinate system for the specification of the BSDF. Like the solid angle, the BRDF can be parameterized using the polar and azimuth angles (θ, ϕ). The polar angle θ is the angle made by a given direction $\vec{\omega}$ with the normal at the observed surface point. The azimuth angle ϕ is the angle between the projection of $\vec{\omega}$ on the tangent space and the tangent at the surface point. Figure 2.4 illustrates the conversion of the unit direction vectors to *angular form.*

The correlation between the unit direction and the angular representation is given by:

$$\begin{gathered} \cos\theta = \vec{\omega} \cdot \vec{N}_x \\ \cos\phi = \vec{\omega} \cdot \vec{T}_x \end{gathered} \tag{2.15}$$

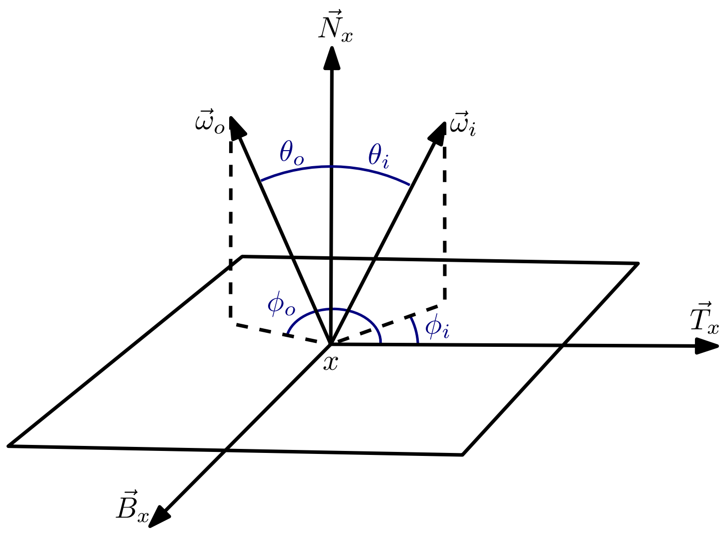


Figure 2.4*: The surface coordinate system defined by the normal $\vec{N}_x$, tangent $\vec{T}_x$ and bitangent $\vec{B}_x$ is used to redefine the directions of the BRDF in terms of pairs of angles.*

The surface coordinate system is a vital convention that must be adopted whenever modelling surface scattering. Hall (1989) draws attention on the necessity of a surface reference system, as well as on the importance of having a mapping between the local surface and the global coordinate systems. The angular parameterization of the BRDF is a local representation, as both directions are defined relative to one of the local surface axes. Conversely, the unit direction parameterization is a global representation, since the incident and exitant directions of the BRDF do not depend on the local axes.

BRDF representation and parameterization are important especially in the context of measured BRDFs. Usually, a real BRDF is sampled at different viewing and lighting angles. The result of the measurement process is a table of reflectance values, which correspond to the observed viewing and lighting angles. This tabular data can be fitted to existing analytic models by using an adequate parameterization. Selecting a good parameterization is essential to the fitting process. One parameterization advantage refers to attaining an increased separability of the BRDF, i.e. the original BRDF can be factorised into distinct bases that can be manipulated independently. Other advantages refer to representing the BRDF more compactly and to reducing its memory footprint.

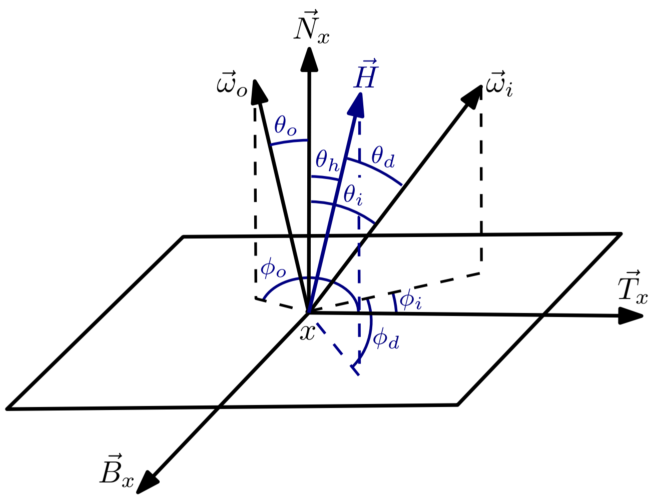


***Figure 2.5**: Parameterization via halfway $\vec{H}(\theta_h, \phi_h)$ and difference $\vec{\omega}_i(\theta_d, \phi_d)$ vectors.*

For example, Rusinkiewicz (1998) proposed the *halfway-difference parameterization*. This representation defines the BRDF in terms of the halfway and difference vectors. The halfway vector is the bisector of the angle described by the incoming and the viewing direction. The difference vector is the incoming vector in a reference frame that establishes the halfway vector as the surface normal. This reference frame can be obtained by means of a Gram-Schmidt orthonormalization (Kautz and McCool 1999):

$$\begin{aligned} \vec{N}'_x &= \vec{H} = \frac{\vec{\omega}_i + \vec{\omega}_o}{\|\vec{\omega}_i + \vec{\omega}_o\|} & \vec{T}'_x &= -\frac{\vec{N}_x - (\vec{N}_x \cdot \vec{H})\vec{H}}{\|\vec{N}_x - (\vec{N}_x \cdot \vec{H})\vec{H}\|} \\ \vec{B}'_x &= \vec{H} \times \vec{T}'_x & \vec{\omega}_i{}' &= [\vec{\omega}_i \cdot \vec{H}, \vec{\omega}_i \cdot \vec{T}'_x, \vec{\omega}_i \cdot \vec{B}'_x] \end{aligned} \tag{2.16}$$

Figure 2.5 illustrates the halfway-difference parameterization relative to the standard normal-tangent-bitangent surface coordinates. The image also displays the polar and the azimuth angles that correspond to the incident and exitant directions of the BRDF.

The main advantage of the halfway-difference parameterization is the separability of the BRDF. The purpose of this representation is to align the retro-reflective and the specular peaks with the axes of the reference frame and reduce their dependence on a combination of axes to mainly one axis. Minimizing the axis dependence of the reflective peaks renders the BRDF separable into different components. Problems arise when the halfway vector lies very close to the surface normal. The specialized literature proposes alternative parameterizations. For instance, Kautz and McCool (1999) discuss the singular value decomposition and the normalized decomposition.

However, most parameterizations are case dependent and have their own advantages and disadvantages. The most problematic aspect of the singular value decomposition is the output of negative values, which are incompliant with the nature of reflectance.

Parameterization is equally important for the efficient representation and storing of BRDF data. Measured BRDFs, which are not fitted to an analytic model, preserve the distinctive features of the original material. However, they represent a dense collection of data, rather than a reflectance model. One disadvantage to using raw data, is the fact that the slightest change in material properties requires new BRDF data. Other concerns regard the indexing, the value retrieval and the storing of the BRDF data. For example, isotropic BRDFs can be stored more compactly due to rotational invariance, i.e. ϕ_o can be ignored. Matusik et al. (2003) used the halfway and difference vectors to render isotropic measured BRDFs, at the cost of a decrease in the accuracy of the data.

2.3.3 Surface reflectance

As a specialization of the BSDF (equation 2.12), the BRDF describes the reflection of light at a surface point and can be used to derive several quantities that characterize the reflectance of a surface. The irradiance field at a surface point enables the deduction of the *reflected radiance* via integration over the incident hemisphere (Jensen 2001, p. 20):

$$L_o(x,\vec{\omega}_o) = \int\limits_{\mathcal{H}_i} f_r(x,\vec{\omega}_i,\vec{\omega}_o)\, dE(x,\vec{\omega}_i) = \int\limits_{\mathcal{H}_i} f_r(x,\vec{\omega}_i,\vec{\omega}_o)\, L_i(x,\vec{\omega}_i)\left|\vec{N}_x\cdot\vec{\omega}_i\right| d\vec{\omega}_i \quad (2.17)$$

Reflectance quantifies the amount of incident light reflected by a surface and is defined as the ratio between the reflected and the incident flux (Nicodemus et al. 1977, p. 8):

$$\rho(x) = \frac{d\Phi_r}{d\Phi_i} = \frac{\int_{\mathcal{H}_r}\int_{\mathcal{H}_i} f_r(x,\vec{\omega}_i,\vec{\omega}_o) L_i(x,\vec{\omega}_i)\left|\vec{N}_x\cdot\vec{\omega}_i\right|\left|\vec{N}_x\cdot\vec{\omega}_o\right| d\vec{\omega}_i d\vec{\omega}_o}{\int_{\mathcal{H}_i} L_i(x,\vec{\omega}_i)\left|\vec{N}_x\cdot\vec{\omega}_i\right| d\vec{\omega}_i} \quad (2.18)$$

where the equivalence $d\Phi_r/dA(x) = \int_{\mathcal{H}_r} L_o(x,\vec{\omega}_o)\left|\vec{N}_x\cdot\vec{\omega}_o\right| d\vec{\omega}_o$ was used to define the reflected flux and the reflected radiance $L_o(x,\vec{\omega}_o)$ was expanded via its definition. Reflectance is also known as the albedo of a material. The flux that is not reflected, is either transmitted via the BTDF or absorbed, so that energy is properly accounted for.

The *bi-hemispherical reflectance* is defined in conditions of isotropic illumination, which imply a constant incident radiance (Mobley et al. 2017). Reflectance reduces to:

$$\rho_{bh}(x) = \frac{1}{\pi} \int_{\mathcal{H}_r} \int_{\mathcal{H}_i} f_r(x, \vec{\omega}_i, \vec{\omega}_o) \left|\vec{N}_x \cdot \vec{\omega}_i\right| \left|\vec{N}_x \cdot \vec{\omega}_o\right| d\vec{\omega}_i d\vec{\omega}_o \tag{2.19}$$

The bi-hemispherical reflectance is known as the Bond, spherical or white-sky albedo.

The *directional-hemispherical reflectance* is obtained by assuming a fixed incident direction (Mobley et al. 2017). This quantity is also known as the black-sky albedo:

$$\rho(x, \vec{\omega}_i) = \int_{\mathcal{H}_r} f_r(x, \vec{\omega}_i, \vec{\omega}_o) \left|\vec{N}_x \cdot \vec{\omega}_o\right| d\vec{\omega}_o \tag{2.20}$$

Nicodemus et al. (1977) analyse multiple quantities that describe surface reflectance.

2.3.4 Local illumination and the rendering equation

When light encounters an obstacle, two types of light interaction can be detected. One type of light interaction takes place at the boundary between two media, whereas the other type occurs as light travels through the intersected object. The boundary/surface interaction is modelled by the BSDF. As it passes through an object, light is either scattered or absorbed. Its exact behaviour is determined by the electrical properties of the traversed object, i.e. by the material character. Metals possess free electrons which resonate when excited and emit electromagnetic radiation when returning to their ground state. Without free electrons, dielectrics have a low and uniform reflectivity. Hall (1989) examines in detail the electrical properties that give a material its character.

Based on the type of incident light, surface interaction can be defined using a direct and an indirect component. The direct component subsumes light reflected/transmitted towards the viewer directly from the light sources. The indirect/scattered component incorporates light arriving at the viewed point from objects other than the light sources. Both components are subjected to attenuation factors, surface microgeometry and Fresnel coefficients (Hecht 2002, p. 114). The attenuation factors can be discarded for environments that lack participating media and comprise objects with constant index of refraction. The geometric factor and the Fresnel coefficients are usually subsumed in the BSDF. The incident light from a given direction $\vec{\omega}_i$ is modelled by the BSDF as:

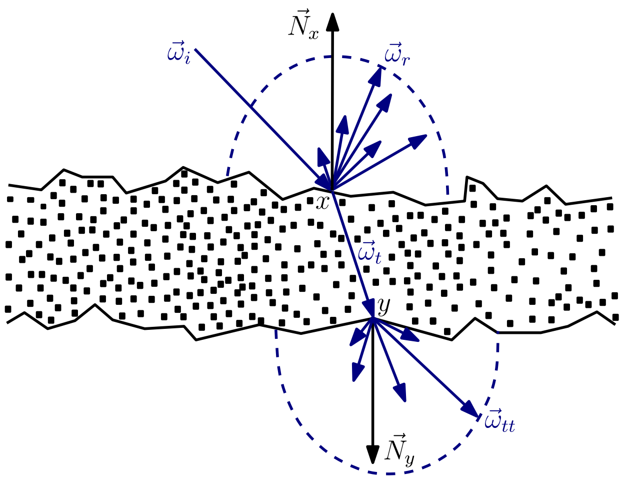


***Figure 2.6**: Boundary reflection (f_r) & transmission (f_t) (amended from Jurohi 2006).*

$$dL_o(x,\vec{\omega}_o) = f_s(x,\vec{\omega}_i,\vec{\omega}_o)L_i(x,\vec{\omega}_i)d\vec{\omega}_i^{\perp}$$

Integrating over incident directions yields the *scattering equation* (Veach 1998, p. 86):

$$L_o(x,\vec{\omega}_o) = \int_{\mathcal{H}} f_s(x,\vec{\omega}_i,\vec{\omega}_o)L_i(x,\vec{\omega}_i)d\vec{\omega}_i^{\perp} \qquad (2.21)$$

where $\mathcal{H}$ denotes either the upper ($\mathcal{H}^+$) or the lower ($\mathcal{H}^-$) hemisphere. Equation (2.21) is also known as the *local illumination model* (Jensen 2001, p. 20). The scattering equation incorporates both the direct and the indirect component, as $\vec{\omega}_i$ may originate at the light sources or at other scene locations. Figure 2.6 shows the direct reflection and transmission at x and the indirect transmission at y caused by the secondary ray $\vec{\omega}_t$.

The scattering equation is generalized by defining the incident radiance based on the exitant one and by adding an emitted component that represents the scene light sources:

$$L_o(x,\vec{\omega}_o) = L_e(x,\vec{\omega}_o) + \int_{\mathcal{H}} f_s(x,\vec{\omega}_i,\vec{\omega}_o)L_o(\tau(x,\vec{\omega}_i),-\vec{\omega}_i)d\vec{\omega}_i^{\perp} \qquad (2.22)$$

where $L_i(x,\vec{\omega}_i) = L_o(\tau(x,\vec{\omega}_i),-\vec{\omega}_i)$ and $\tau(x,\vec{\omega}_i)$ is the *tracing/ray-casting function*, which casts a ray from x in the direction $\vec{\omega}_i$ and determines the first scene intersection:

$$\tau(x,\vec{\omega}_i) = x + \alpha_{min}(x,\vec{\omega}_i)\vec{\omega}_i \qquad (2.23)$$

$\alpha_{min}(x,\vec{\omega}_i)$ represents the *boundary distance function* defined as (Veach 1998, p. 110):

$$\alpha_{min}(x,\vec{\omega}_i) = \inf\{\alpha > 0 \mid x + \alpha\vec{\omega}_i \in \mathcal{M}\} \qquad (2.24)$$

which yields the first point on the *set of scene surfaces* $\mathcal{M}$ visible from x along $\vec{\omega}_i$. Equation (2.22) completely defines the light transport problem and it is known as *the rendering equation* (Kajiya 1986) or *the light transport equation* (Veach 1998, p. 90).

2.4 Light transport

Light transport theory regards the scattering of light in a synthetic environment. The rendering equation (Kajiya 1986) expresses the incident radiance as the radiance that arrives at the observed point from another scene location. Being of a scattered nature, radiance can be evaluated recursively at distinct scene locations. The rendering equation is appliable to environments without participating media. In vacuum, radiance is constant along a propagation direction and its exitant component must equal the sum between the emitted and the scattered component. Even so, the rendering equation is complex and it entails an infinite series, which can only be evaluated numerically by the existing global illumination algorithms. Consequently, various algorithms evaluate the rendering equation recursively, based on different assumptions and/or constraints.

2.4.1 The light transport linear operators

As expressed by the rendering equation, light transport is a two-step process. Surface interaction is defined by the scattering equation. From the impact point light propagates further through reflections and transmissions. The rendering equation captures the notion of propagation in the equality between the incident and the exitant radiance, i.e. $L_i(x, \vec{\omega}_i) = L_o(\tau(x, \vec{\omega}_i), -\vec{\omega}_i)$. Without participating media, radiance is constant along a propagation direction and thus it can be represented only as it exits a surface, without considering every point of its trajectory. That is, the radiance that exits a surface is the radiance that reaches the first surface intersected along the propagation direction. The known implication is that the incident and the exitant radiance can be interconverted.

Veach (1998) uses the *radiance function* to describe the distribution of radiance in a scene. The radiance function simply associates radiance values to position-direction pairs. A position-direction pair $(x, \vec{\omega})$ can be interpreted as a ray that has origin x and direction $\vec{\omega}$. The entirety of such rays forms the ray space $\mathcal{R} = \mathcal{M} \times \mathcal{D}$, which is the Cartesian product between the set of scene surfaces $\mathcal{M}$ and the set of unit directions $\mathcal{D}$.

Using the ray space abstraction, the radiance function can be defined as $L : \mathcal{R} \rightarrow \mathbb{R}$. Though the negative values of the codomain cannot represent radiance values, they are allowed so that the radiance functions can form a space. Veach (1998, p. 107) studies the properties which render function spaces desirable, especially for analysis purposes.

The light transport linear operators work on the space of radiance functions and interconvert the incident and the exitant radiance. A *linear operator* is a linear mapping between vector spaces. Scattering and propagation are defined by two linear operators.

The *scattering operator* maps the incident radiance function to its exitant homologue:

$$(KL_i)(x, \vec{\omega}_o) = \int_{\mathcal{D}} f_s(x, \vec{\omega}_i, \vec{\omega}_o)\, L_i(x, \vec{\omega}_i) d\vec{\omega}_i^{\perp} \tag{2.25}$$

The scattering operation can be concisely expressed as $L_o = KL_i$ (Veach 1998, p. 110).

The *propagation operator*, also known as the *geometric operator*, defines the reverse transformation in that it maps the exitant radiance function to the incident counterpart:

$$(\mathcal{P}L_o)(x, \vec{\omega}_i) = \begin{cases} L_o(\tau(x, \vec{\omega}_i), -\vec{\omega}_i) & if \;\; \alpha_{min}(x, \vec{\omega}_i) < \infty \\ 0 & otherwise \end{cases} \tag{2.26}$$

where $\tau(x, \vec{\omega}_i)$ is the tracing function and $\alpha_{min}(x, \vec{\omega}_i)$ is the boundary distance function. When $\alpha_{min}(x, \vec{\omega}_i) = \infty$, $\mathcal{M}$ is not intersected and τ remains undefined (Veach 1998, p. 110). The rectilinear propagation of light can be abridged as $L_i = \mathcal{P}L_o$.

The scattering and the propagation operator represent one step in the light transport process. Their combination yields the *light transport operator* $T = K\mathcal{P}$, which maps the exitant radiance onto itself, in a single scattering step: $TL_o = K(\mathcal{P}L_o) = KL_i = L_o$.

Using the light transport operator, the rendering equation (2.22) can be formulated as:

$$L_o = L_e + TL_o \tag{2.27}$$

Its *analytic solution* can be obtained by inverting the light transport operator, as shown:

$$(\mathcal{J} - T)L_o = L_e$$

$$L_o = (\mathcal{J} - T)^{-1}L_e$$

$$L_o = L_e + TL_e + T^2L_e + T^3L_e + \cdots = \sum_{i=0}^{\infty} T^i L_e \tag{2.28}$$

where $\mathcal{J}$ is the identity operator that leaves the radiance function unchanged. The expansion in deduction (2.28) is the Neumann series for the equilibrium radiance. This series emphasizes two important properties of the rendering equation. Practically, the rendering equation cannot be solved in closed-form. Hence, all the global illumination algorithms estimate it with more or less stringent constraints. The physical significance of the rendering equation is the progressive scattering of light through an environment.

The analytic solution to the rendering equation is a mathematical formalism with several implications. Firstly, the numerical techniques used to estimate the rendering equation determine the constraint level an algorithm assumes on various light transport components, like illumination, surface reflectance models or geometric representation. Secondly, the scattering and propagation operators are mechanisms of light transport that clarify the relationship between the incident and the exitant quantities. Lastly, all the numerical solutions that estimate the rendering equation have a physical substrate.

2.4.2 The surface domain formulation

The rendering equation was defined using the concepts of ray and ray-casting. The ray-casting function is central to an entire class of global illumination algorithms. Its purpose is to determine the closest ray-scene intersections. The punctiform nature of the ray-casting function abstracts the reliant algorithm from the input complexity. Light sources, surface models, geometric representation and object interrelations are decoupled from the algorithm and can be treated generically. Still, object interrelations detail the structure of the rendering equation and define a common basis for various algorithms. Two steps detail surface interrelations (Pharr and Humphreys 2004, p. 737).

The first step redefines the radiance function and the BDSF using surface locations. Veach (1998, p. 106) redefines the domain of the radiance function as a Cartesian square on the set of scene surfaces $\mathcal{M}$ and recasts the radiance function as $L : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$. Using this reformulation, each pair $(x, \vec{\omega})$ can be expressed as $x \rightarrow x'$. A notable difference between the two notations is that $\vec{\omega}$ may indicate a direction towards infinity ($\alpha_{min} = \infty$), while $x \rightarrow x'$ specifies only directions between concrete surface locations. Hence, the surface-based representation of the radiance function and of the BSDF is:

$$L(x \rightarrow x') = L(x, \vec{\omega})$$
$$and$$
$$f_s(x \rightarrow x' \rightarrow x") = f_s(x', \vec{\omega}_i, \vec{\omega}_o)$$

where $\vec{\omega} = \widehat{x' - x}$, $\vec{\omega}_i = \widehat{x - x'}$ and $\vec{\omega}_o = \widehat{x" - x'}$ are unit vectors with upward directions. The hat operator represents a unit-length vector and it is formally defined as follows:

$$\hat{u} = \frac{u}{||u||} \tag{2.29}$$

where $||u||$ is the length of vector u. Throughout the current work the hat operator will be used to construct unit-length vectors that point from one surface location to another.

The second step, entails a change in the integration variable. The rendering equation must be converted from an integral over directions to one over surface areas. To this end, the solid angle must be related to the surface area through the following equation:

$$d\vec{\omega}_i = \frac{|\vec{N}_x \cdot \widehat{x' - x}|dA(x)}{||x - x'||^2} = \frac{|cos\theta|dA(x)}{||x - x'||^2} \tag{2.30}$$

The rendering equation uses the projected solid angle as an integration variable. Hence, the projected differential solid angle is obtained by multiplying equation (2.30) with the cosine of the angle made by the incident direction $x \rightarrow x'$ with the normal at x':

$$d\vec{\omega}_i^{\perp} = \frac{|\vec{N}_{x'} \cdot \widehat{x - x'}||\vec{N}_x \cdot \widehat{x' - x}|}{||x - x'||^2} dA(x) = \frac{|cos\theta'||cos\theta|}{||x - x'||^2} dA(x) \tag{2.31}$$

Equation (2.31) can be applied only when the points x and x' are mutually visible. The visibility between two points is determined through the binary function $V(x \leftrightarrow x')$, which associates visibility with a value of 1 and occlusion with a value of 0. Coupling the visibility function with the ratio from equation (2.31), yields the *geometric factor*:

$$G(x \leftrightarrow x') = V(x \leftrightarrow x') \frac{\left|\vec{N}_{x'} \cdot \widehat{x - x'}\right|\left|\vec{N}_x \cdot \widehat{x' - x}\right|}{\|x - x'\|^2} = V(x \leftrightarrow x') \frac{|cos\theta'||cos\theta|}{\|x - x'\|^2} \quad (2.32)$$

The geometric factor represents the change of variable. Applying the conversion steps, yields the *surface form* of the rendering equation (Pharr and Humphreys 2004, p. 738):

$$L_o(x' \to x") = L_e(x' \to x") + \int_{\mathcal{M}} f_s(x \to x' \to x") L_o(x \to x') G(x \leftrightarrow x') dA(x) \quad (2.33)$$

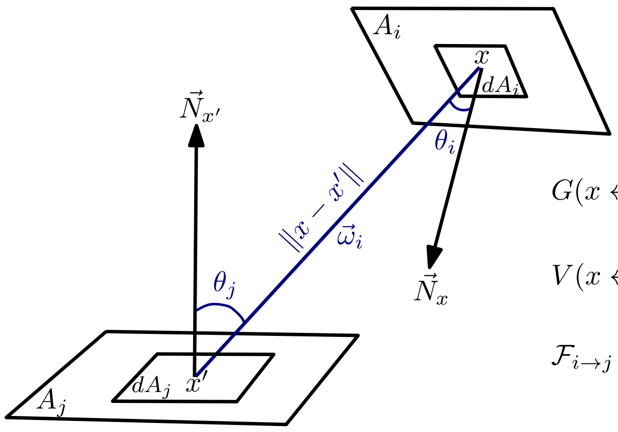
Another designation for equation (2.33) is the *three-point form* (Veach 1998, p. 221).

One global illumination algorithm, which uses extensively the surface interrelations, is radiosity (Goral et al. 1984). Radiosity assesses the energy equilibrium for purely diffuse environments. As only diffuse surfaces are considered, the angular dependence of the radiance and the BSDF vanishes and only the radiosity exchange between sets of patches is computed. Consequently, equation (2.33) reduces to the ensuing equation:

$$B(x') = B_e(x') + \frac{\rho_d(x')}{\pi} \int_{\mathcal{M}} B(x)\, G(x \leftrightarrow x') dA(x) \quad (2.34)$$

where B is the radiosity defined in equation (2.7) and ρ_d is the diffuse reflectance (Jensen 2001, p. 21). Figure 2.7 expresses the form factor between two diffuse patches.

Other algorithms which benefit from the explicit surface interrelations are the ones designated as Monte Carlo ray tracing. These algorithms rely on the path integral framework to compute the numerical light transport solution. The path integral framework expands the three-point rendering equation (2.33) to an integral over paths.



$$G(x \leftrightarrow x') = V(x \leftrightarrow x') \frac{|\vec{N}_x \cdot (-\vec{\omega}_i)||\vec{N}_{x'} \cdot \vec{\omega}_i|}{\|x-x'\|^2}$$

$$V(x \leftrightarrow x') = \begin{cases} 1 & \text{if } x \textit{ is visible from } x' \\ 0 & \text{if } x \textit{ is occluded for } x' \end{cases}$$

$$\mathcal{F}_{i \to j} = \frac{1}{\pi A_i} \iint G(x \leftrightarrow x') dA_i dA_j$$

Figure 2.7: *The form factor $\mathcal{F}_{i\to j}$ used to compute the radiosity from patch i to patch j.*

2.4.3 The path integral formulation

The goal of a light transport algorithm is to produce a set of measurements $\mathcal{I}_0, \dots, \mathcal{I}_{\wp}$ that form the image of a 3D model. Like a real-world camera, a global illumination algorithm generates the output image by measuring the responsivity of a matrix of hypothetical sensors to the radiance incident on it. Each measurement $\mathcal{I}_j,\ 0 \le j \le \wp$, represents a pixel value and describes the sensor responsivity associated with the pixel:

$$\mathcal{I}_j = \int_{\mathcal{R}} W_e^j(x,\vec{\omega}_i) L_i(x,\vec{\omega}_i)\, dA(x) d\vec{\omega}_i^{\perp} = \int_{\mathcal{R}} W_e^j(x,\vec{\omega}_i)\, d^2\Phi_i \qquad (2.35)$$

where $W_e^j(x,\vec{\omega}_i)$ is the sensor *responsivity* for pixel j. Known as the *measurement equation* (Veach 1998, p. 89), equation (2.35) integrates responsivity over sensor area.

Veach (1998, p. 91) treats sensor responsivity as an emitted quantity and defines an equation which specifies the importance attributed to the light that reaches a sensor:

$$W_o(x,\vec{\omega}_o) = W_e(x,\vec{\omega}_o) + \int_{\mathcal{H}} f_s(x,\vec{\omega}_o,\vec{\omega}_i) W_o(\tau(x,\vec{\omega}_i), -\vec{\omega}_i) d\vec{\omega}_i^{\perp} \qquad (2.36)$$

Named the *importance transport equation*, equation (2.36) defines the equilibrium for the *importance function* $W : \mathcal{R} \to \mathbb{R}$, whose values correspond to sensor responsivity measurements. Like the radiance function, the importance function has a real codomain, which ensures that the properties associated with function spaces are entirely preserved.

The importance transport equation evaluates the equilibrium importance by applying the BSDF with exchanged directional parameters. The BSDF is not symmetric and the transposition of the directions $\vec{\omega}_i$ and $\vec{\omega}_o$ yields the *adjoint BSDF* (Veach 1998, p. 93):

$$f_s^*(x, \vec{\omega}_i, \vec{\omega}_o) = f_s(x, \vec{\omega}_o, \vec{\omega}_i) \tag{2.37}$$

The adjoint BSDF is a pivotal concept, especially for bidirectional algorithms, as it evaluates importance. The BSDF models the scattering of light by a surface. Importance flows in the opposite direction of light and thus it requires that the BSDF be adapted to reflect a scattering process in the reversed direction of the light flow. Hence, the BSDF $f_s(x, \vec{\omega}_i, \vec{\omega}_o)$ evaluates radiance, whereas the adjoint BSDF $f_s^*(x, \vec{\omega}_i, \vec{\omega}_o)$ evaluates importance. In terms of propagation, if $\vec{\omega}_i$ is always the sampled direction then, the BSDF scatters importance particles and the adjoint BSDF scatters photons.

Using the linear operators, the importance transport equation can be reformulated as:

$$W_o = W_e + K^*\mathcal{P}W_o \tag{2.38}$$

where K^* is the *adjoint scattering operator*, which differs from the scattering operator K in using the adjoint BSDF, i.e. $(K^*W_i)(x, \vec{\omega}_o) = \int_{\mathcal{D}} f_s^*(x, \vec{\omega}_i, \vec{\omega}_o)\, W_i(x, \vec{\omega}_i) d\vec{\omega}_i^{\perp}$. The geometric operator is self-adjoint, since the first intersection point from which incident particles arrive does not change with the evaluated quantity. That is, $\mathcal{P}$ converts radiance and importance in the same way $L_i = \mathcal{P}L_o$ and $W_i = \mathcal{P}^*W_o = \mathcal{P}W_o$. Consequently, the measurement equation can be restated as shown in the following:

$$I_j = \int_{\mathcal{R}} W_e\mathcal{P}(\mathcal{I} - T)^{-1}L_e \asymp \int_{\mathcal{R}} \mathcal{P}(\mathcal{I} - K^*\mathcal{P})^{-1}W_eL_e \tag{2.39}$$

where $T_W = K^*\mathcal{P}$ is the *importance transport operator*, $L_o = KL_i$ and $W_o = K^*W_i$. Veach (1998) minutely discusses the adjoint transport principles briefly presented here.

The goal of the path integral formulation is to express the measurement equation as an integral over the space of paths. The three-point rendering equation implies a recursive sampling of the scene, one surface at a time. The path integral formulation expands the three-point rendering equation by considering a sequence of surface scattering events. That is, surfaces are interrelated through the explicit modelling of the light flow.

Mathematically, the measurement equation can be redefined on the space of paths as:

$$I_j = \int_{\mathfrak{X}} f_j(\bar{x}) d\mu(\bar{x}) \tag{2.40}$$

where $\bar{x} = x_0 \dots x_k$ is a path of length k and $\mathfrak{X}$ is the set of paths of all finite lengths:

$$\mathfrak{X} = \bigcup_{k=1}^{k<\infty} \mathfrak{X}_\mathrm{k} \tag{2.41}$$

$d\mu(\bar{x})$ is the *differential area-product measure* defined over $x_i \in \mathcal{M}$, with $0 \le i \le k$:

$$d\mu(\bar{x}) = dA(x_0) \dots dA(x_k) \tag{2.42}$$

The *path integral formulation* (2.40) can be fully derived by expanding equation (2.33):

$$I_j = \sum_{k=1}^{k<\infty} \int_{\mathcal{M}^{k+1}} L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \\ W_e^j(x_{k-1} \to x_k) dA(x_0) \dots dA(x_k) \tag{2.43}$$

The integrand in equation (2.43) can be abstracted in the *measurement contribution function* f_j (Veach 1998, p. 223). For a path of length k, f_j is defined as shown below:

$$f_j(\bar{x}) = L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \\ \prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \, W_e^j(x_{k-1} \to x_k) \tag{2.44}$$

The physical interpretation of the path integral formulation is that of light scattering through an environment as many times as indicated by the length of the sampled path. For a path $\bar{x} = x_0 \dots x_4$, the measurement contribution function is defined as follows:

$$\begin{aligned} f_j(\bar{x}) \quad = \quad & L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) f_s(x_0 \to x_1 \to x_2) \\ & G(x_1 \leftrightarrow x_2) f_s(x_1 \to x_2 \to x_3) G(x_2 \leftrightarrow x_3) \\ & f_s(x_2 \to x_3 \to x_4) G(x_3 \leftrightarrow x_4) W_e^j(x_3 \to x_4) \end{aligned}$$

Figure 2.8 shows the terms of the measurement contribution function for $\bar{x} = x_0 \dots x_4$.

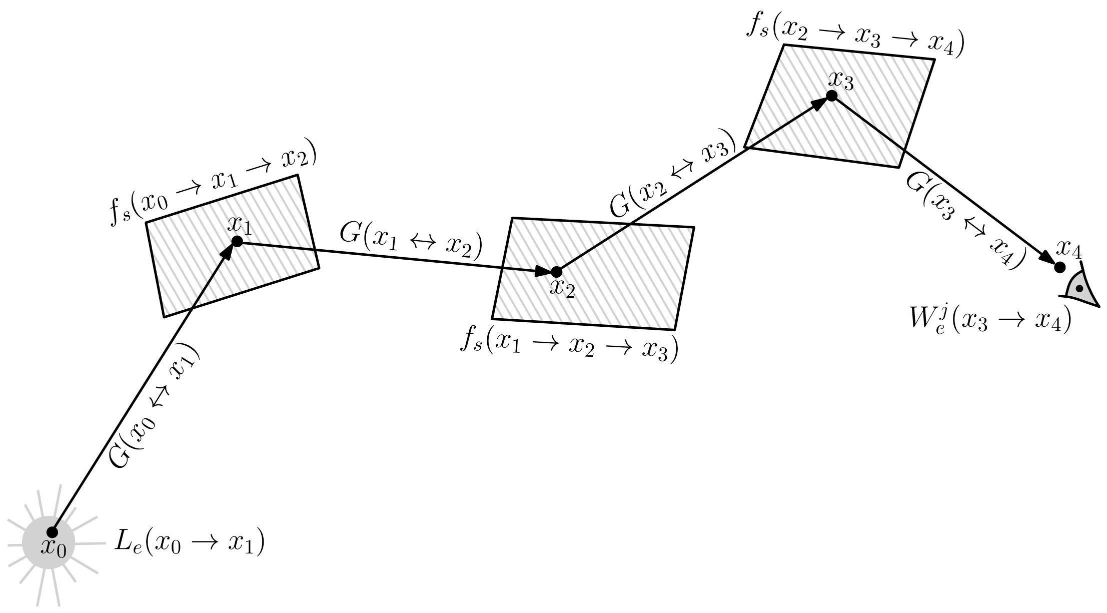

$f_s(x_2 \to x_3 \to x_4)$
x_3
$G(x_2 \leftrightarrow x_3)$
$f_s(x_0 \to x_1 \to x_2)$
x_1
$G(x_1 \leftrightarrow x_2)$
$G(x_3 \leftrightarrow x_4)$
x_4
x_2
$W_e^j(x_3 \to x_4)$
$f_s(x_1 \to x_2 \to x_3)$
$G(x_0 \leftrightarrow x_1)$
x_0
$L_e(x_0 \to x_1)$

***Figure 2.8**: Emission, scattering and measurement events on a path of length $k = 4$.*

The path integral formulation has several advantages. One advantage is the explicit modelling of the emission, scattering and measurement events along full transport paths. Instead of focusing on a quantity scattered between two surfaces, the path integral formulation operates on the geometric construct of path, with the result that a higher degree of generality is obtained. Equation (2.43) abstracts the dichotomy between light and importance particles. The greatest advantage of this formulation is the ability to construct paths arbitrarily, from the camera, a light source or any other scene object. The flexibility of arbitrarily constructing paths underlies the development of sampling techniques and of Monte Carlo solutions. Multiple importance sampling (Veach and Guibas 1995), bidirectional path tracing (Veach and Guibas 1994) and Metropolis light transport (Veach and Guibas 1997) rely on the path integral framework.

2.4.3.1 The throughput measures

The path integral formulation constitutes a natural framework for the definition of several path space measures. Generally known as *throughput*, each measure has its own physical meaning. Veach (1998, p. 243) distinguishes four throughput measures.

The cone of directions subsumed in a unit solid angle around a ray $\mathcal{r}(x, \vec{\omega})$, impinges on a unit surface area dA with an energy proportional to the ensuing product measure:

$$d\mu(\mathcal{r}) = d\mu(x, \vec{\omega}) = dA(x) d\vec{\omega}^{\perp} \tag{2.45}$$

The *differential throughput*, $d\mu(\mathcal{r})$ measures the light-carrying capacity of a small cone of rays. The *throughput* of a non-infinitesimal cone of rays $\mathcal{D}_{\mathcal{r}}$ can be obtained by simply integrating the differential throughput measure $d\mu$ over the domain $\mathcal{D}_{\mathcal{r}}$, giving:

$$\mu(\mathcal{D}_{\mathcal{r}}) = \int\limits_{\mathcal{D}_{\mathcal{r}}} dA(x) d\vec{\omega}^{\perp} \tag{2.46}$$

The *measurement contribution throughput* quantifies the amount of light that a set of paths $\mathcal{D}_{\bar{x}}$ brings to a measurement I_j. It is obtained by integrating the measurement contribution function $f_j(\bar{x})$ with respect to the differential area-product measure $d\mu(\bar{x})$:

$$\mu_m^j(\mathcal{D}_{\bar{x}}) = \int\limits_{\mathcal{D}_{\bar{x}}} f_j(\bar{x})\, d\mu(\bar{x}) \tag{2.47}$$

This throughput is a fundamental measure used to derive the other throughput measures.

The *power throughput* is obtained by omitting the importance function W_e^j from $f_j(\bar{x})$:

$$\mu_p^k(\mathcal{D}_{\bar{x}}) = \int\limits_{\mathcal{D}_{\bar{x}}} L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1})\, d\mu(\bar{x}) \tag{2.48}$$

As the measurement contribution function is dependent on the path length, $\mu_p^k(\mathcal{D}_{\bar{x}})$ quantifies the power carried by a set of k-length paths. The length-independent measure quantifies the power within the entire scene and is defined as shown in the following:

$$\mu_p(\mathcal{D}_{\bar{x}} \subset \mathfrak{X}) = \sum_{k=1}^{k<\infty} \mu_p^k(\mathcal{D}_{\bar{x}} \subset \mathfrak{X}_k) \tag{2.49}$$

The *scattering throughput* assumes uniform light source emission and thus is defined by extracting the emitted radiance function from the integrand of the power throughput:

$$\mu_s^k(\mathcal{D}_{\bar{x}}) = \int\limits_{\mathcal{D}_{\bar{x}}} G(x_0 \leftrightarrow x_1) \prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1})\, d\mu(\bar{x}) \tag{2.50}$$

The scattering throughput indicates how the power-carrying capacity of a set of paths is modulated by the geometry and the surface properties of a given environment. The length-independent scattering throughput is defined similarly to the power analogue μ_p.

The *geometric throughput* assumes both uniform light source emission and perfectly random scattering. The latter condition entails a constant BSDF over the hemisphere:

$$f_s(x_{i-1} \rightarrow x_i \rightarrow x_{i+1}) = \frac{1}{2\pi}$$

Hence, the geometric throughput further decouples the power-carrying capacity of a set of paths from the surface properties. Like the other measures, it is length-dependent:

$$\mu_g^k(\mathcal{D}_{\bar{x}}) = \left(\frac{1}{2\pi}\right)^{k-1} \int_{\mathcal{D}_{\bar{x}}} G(x_0 \leftrightarrow x_1) \dots G(x_{k-1} \leftrightarrow x_k) d\mu(\bar{x}) \qquad (2.51)$$

The length-independent counterpart is defined similarly to the power throughput μ_p and it constitutes a generalization of the throughput measure defined in equation (2.46). Veach (1998, p. 245) remarks that scenes with finite surface area do not entail a finite geometric throughput, since the geometric factors (equation 2.32) do not absorb energy as light propagates along the sampled paths. Without participating media, absorption occurs at the BSDF level. To determine the extent to which a set of paths is modulated by the surface properties or by the geometry, the ratio $\mu_s(\mathcal{D}_{\bar{x}})/\mu_g(\mathcal{D}_{\bar{x}})$ can be evaluated.

2.5 Light transport algorithms

In the absence of participating media, the light transport problem is entirely described by the rendering equation (2.22). If the light transport linear operators are the mathematical tools used to derive the analytic solution of the rendering equation, in practice the light transport algorithms can only provide numerical solutions. The approximative nature of various global illumination algorithms stems from the limited resources and from the infiniteness entailed by the Neumann series (equation 2.28).

The literature abounds in techniques which compute global illumination based on a set of resource constraints (Ritschel et al. 2012; Davidovič et al. 2014). From the early stages of the current work, an important prerequisite for the development of the light transport framework (chapter 4) was the ability to simulate a wide range of illumination effects. This imperative represented the starting point and the canon used to identify the algorithm that would act as a development foundation and steppingstone towards the set goal. The goal is to support scene dynamism for high-quality light transport simulations. The current section reflects the technical review carried for Optis, which led to the decisions that shaped the development of the path manipulation algorithm.

Though inappropriate for the set objectives, the classes of radiosity and precomputed light transport algorithms were part of the contextual analysis and contributed towards making an informed decision. Due to its generality, the class of point-based sampling algorithms represented the strongest candidate for the development of the light transport framework. Having identified an appropriate algorithmic basis, section 2.6 analyses the current work relative to the recent advances in light transport simulation.

2.5.1 Radiosity

Radiosity (Goral et al. 1984) was presented in subsection 2.4.2 as one of the algorithms which extensively use surface interrelations. Radiosity computes a view-independent, global illumination solution and can be credited with the introduction of physically-based simulation to image synthesis. The algorithms in this class originated from the field of radiative heat transfer, where they were designated as *finite element methods*. However, the original algorithm (Goral et al. 1984) is restrictive and computationally expensive. Firstly, the form factors are computed between each pair of scene patches, with the result that the computational cost increases with the complexity of the scene. For a scene discretized into N patches, radiosity requires $O(N^2)$ memory to store the matrix of form factors and $O(N^3)$ time to solve the equations via Gaussian elimination. Secondly, the radiosity solution depends on the tessellation of the scene. When the level of tessellation is inadequate, the illumination solution suffers from artefacts, such as blurred shadow edges. Lastly, radiosity cannot generate effects other than diffusion.

Research in this area brought many improvements to the original algorithm. Immel et al. (1986) extended the original radiosity algorithm to account for glossy reflections. Building on hierarchical radiosity (Hanrahan et al. 1991), Smits et al. (1994) proposed a clustering algorithm that reduces both the form factors and the visibility calculations.

Still, many radiosity algorithms are affected by severe constraints. The dependence on the mesh tessellation is the major limitation in accurately simulating light transport. The previously cited advances display limitations that either increase the complexity of the algorithm or generate artefacts. Immel et al. (1986) use the concept of hemicube to compute patch visibility. The hemicube introduces additional operations, like the projection of the environment on each of the cubes that surround the point samples of a patch. The approximation of the directional hemisphere with the hemicube and the interpolation of the radiance among exitant directions produce discretization artefacts. Figure 2.9 depicts the projection operation on the hemicube centred on a patch sample.

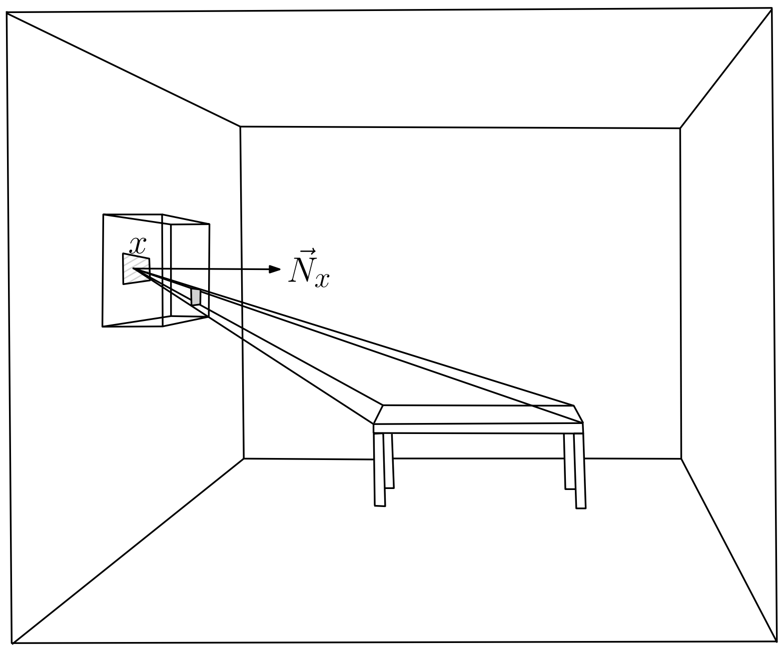


***Figure 2.9**: The projection of the environment on the imaginary hemicube centred on the point sample x of one of the scene patches (amended from Immel et al. 1986, p. 4).*

The clustering algorithm for hierarchical radiosity (Smits et al. 1994), uses ray tracing to supplant certain operations. Specifically, the hemicube construct is replaced by ray casting, so that visibility may be computed in a much simpler and efficient manner. However, the main disadvantage remains the mesh tessellation. The actual rendering step processes a different mesh from the one used during the computation of the global solution. Having two separate meshes, implies at least the recomputation of the form factor between the source and the target point, if not the recalculation of their visibility.

Interactive radiosity algorithms also restrict the simulation process through various assumptions. One end of the assumption spectrum refers to limiting the types of objects that can be present in a scene. Dachsbacher et al. (2007) construct their algorithm by exclusively accepting opaque solids. The other end of the assumption spectrum refers to defining and using hypothetical constructs, which do not have exact correspondents in the rendered scene. Dachsbacher et al. (2007) define a linear operator based on a hypothetical BTDF, which does not have a physical correspondent in the used scene.

Such assumptions and the input dependence, excluded radiosity from being deemed a potential algorithmic foundation for the development of the light transport framework.

2.5.2 Precomputed light transport

Precomputed light transport is a class of global illumination algorithms, described by two phases. The pre-processing phase derives the essential light transport information, so that the rendering phase may benefit from interactive or even real-time framerates.

Most precomputed light transport algorithms use a light transport operator to model the light interactions between object patches. The light transport operator is defined like the scattering operator (equation 2.25), but also includes the visibility function V:

$$\begin{aligned}(\mathcal{T}L_i)(x,\vec{\omega}_o) &= \int_{\mathcal{D}} f_s(x,\vec{\omega}_i,\vec{\omega}_o)\, V(x,\vec{\omega}_i)\left|\vec{N}_x\cdot\vec{\omega}_i\right| L_i(x,\vec{\omega}_i)d\vec{\omega}_i \\ &= \int_{\mathcal{D}} \Gamma(x,\vec{\omega}_i,\vec{\omega}_o)L(x,\vec{\omega}_i)d\vec{\omega}_i \end{aligned} \tag{2.52}$$

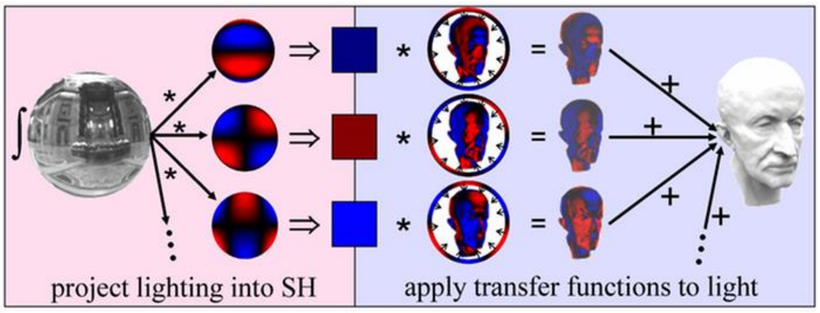
The *transport function* Γ absorbs the BSDF, the visibility function and the cosine term.

The *pre-processing phase* projects both the radiance and the transport function onto orthonormal bases. The result of the projection operation is a set of coefficients, which modulates the basis functions and reconstructs the original radiance/transport function.

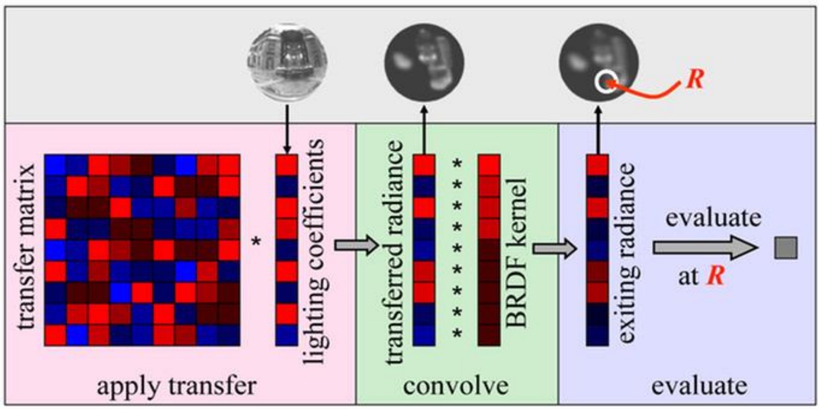
The *rendering phase* involves the integration of the product between the radiance and the transport function. Using the decomposed illumination and transport information, the reflected radiance, at each visible point, is simply computed as the inner product between the vector of radiance coefficients and the vector of transport coefficients:

$$\begin{aligned} L_o(x,\vec{\omega}_o) &= \int_{\mathcal{H}} \left(\sum_k L_k \mathscr{b}_k(\vec{\omega}_i)\right)\left(\sum_n \Gamma_n(x,\vec{\omega}_o)\mathscr{b}_n(\vec{\omega}_i)\right)d\vec{\omega}_i \\ &= \sum_k\sum_n L_k\Gamma_n(x,\vec{\omega}_o)\int_{\mathcal{H}} \mathscr{b}_k(\vec{\omega}_i)\mathscr{b}_n(\vec{\omega}_i)d\vec{\omega}_i \\ &= \sum_k\sum_n L_k\Gamma_n(x,\vec{\omega}_o)\delta_{kn}(\vec{\omega}_i) = \sum_k L_k\Gamma_k(x,\vec{\omega}_o) \\ &= L\cdot\Gamma \end{aligned} \tag{2.53}$$

where L_k is a radiance coefficient, $\mathscr{b}_k$ is a radiance basis function, Γ_n is a transport coefficient, $\mathscr{b}_n$ is a transport basis function and lastly δ_{kn} is the Kronecker delta.



diffuse surface self-transfer



glossy surface self-transfer

***Figure 2.10**: Self-transfer rendering for diffuse & glossy objects (Sloan et al. 2002, p. 2).*

The above equation computes only direct illumination and light source occlusion. Each higher-order term of the Neumann series (equation 2.28) requires a similar derivation. Moreover, the above mechanism can be applied only under the assumption of distant illumination. The low frequency nature of distant light, reduces the dimensionality of the radiance function and with it the light transport computational complexity. Uniform illumination does not vary with position, but depends solely on the incident direction.

Precomputed radiance transfer (Sloan et al. 2002) uses this assumption to render the self-shadowing and the interreflection pattern of a diffuse or glossy object. For diffuse objects, the dimensionality of the transport function reduces as well. Because a purely diffuse BRDF is uniform over the surface, the dependence on the exitant direction can be eliminated. Thus, diffuse object rendering reduces to a simple dot product between the radiance and the transport coefficients vectors. Rendering glossy objects requires additional computations, since the transport function preserves its dimensionality. Even so, the admissible glossy surfaces are limited to rotationally invariant BRDFs. Figure 2.10 compares the rendering of self-transfer between diffuse and glossy objects.

By computing the light transfer between points located on distinct objects, precomputed radiance transfer (Sloan et al. 2002) can simulate the effect of an object on its vicinity, i.e. inter-object illumination. However, the influence of multiple objects on the same receiver was acknowledged as computationally intractable, despite the simplifying assumption of constant lighting across the neighbourhoods of the influencing objects.

The premises of distant illumination and radially symmetric surface definitions cause the high frequency details (e.g. sharp shadows, specularity and caustics) to be blurred.

Kautz et al. (2002) remove the limitation on surface definitions by processing generic BRDFs. Without the assumption of rotational invariance, the spherical harmonics coefficients of the BRDF and the spherical harmonics projection of the incident light are both computed during the pre-processing phase. The rendering phase still computes a simple dot product between the light and the BRDF coefficients vectors. However, the assumption of low frequency illumination is maintained. Moreover, precomputed radiance transfer is used to generate the self-shadowing and interreflection patterns.

Ng et al. (2004) achieve all frequency relighting in conditions of distant illumination. The sharp illumination effects are retained by computing the integral of three projected factors. The pre-processing phase projects the scattering and the visibility function onto the Haar wavelet basis. For each frame, the radiance function is projected onto the Haar basis and is non-linearly approximated with the N largest wavelets. The product between the light, the scattering and the visibility coefficients is accumulated at each point. The triple product wavelet approach removes the low pass filter effect. Still, the algorithm fails to render anisotropic BRDFs and higher-order light bounces.

Wang et al. (2005) extend precomputed light transport to render different types of illumination effects, such as translucency. Nevertheless, effects like transparency or refracted caustics are not supported. Moreover, the assumption of distant illumination is preserved to the detriment of other illumination models. Precomputed light transport achieves high performance, at the cost of input generality and illumination effects.

2.5.3 Ray tracing

Ray tracing is one of the most general classes of global illumination algorithms. Its generality stems from the punctiform nature of the ray-casting function, whose role is to trace rays and determine their closest intersections with the scene geometry. The determined intersections are called *point samples*. Because ray tracing algorithms use point sampling to estimate the illumination of a scene, they are also known as *point sampling algorithms*. Their major advantage is the abstraction from input complexity. Compared to radiosity and precomputed light transport, the ray tracing algorithms avoid the contrivances related to geometric representation, incident illumination and surface properties. All components are treated generically and without precomputation.

Recursive ray tracing (Whitted 1980) was the algorithm which popularized ray tracing. Together with radiosity, ray tracing can be credited with advancing the synthesis of realistic images towards physically-based light transport simulations.

A *classic ray tracing algorithm* computes the average radiance of each pixel, by tracing several *primary rays* through the pixel. The primary rays carry the radiance reflected by the first determined point samples to the observer. At every primary sample, the direct illumination is computed via the scattering equation (2.21). A point sample is illuminated by a light source only if the *shadow ray* traced between them is unobstructed by other objects, i.e. the visibility function for the shadow ray returns 1. From the primary samples, *secondary rays* are generated and traced in reflected or transmitted directions, according to the surface BSDFs. The radiance of the secondary samples is computed analogously to the primary samples, via the scattering equation.

Recursive ray tracing (Whitted 1980) extends the evaluation of the scattering equation to higher-order specular rays. From the primary samples, rays are recursively traced through various specularly reflected or transmitted bounces, as indicated by the surface BSDFs. At each impact point, radiance is computed using the scattering equation. It is precisely this recursive evaluation that gave the algorithm its name. Jensen (2001, p. 37) provides the pseudocode for the recursive ray tracing algorithm.

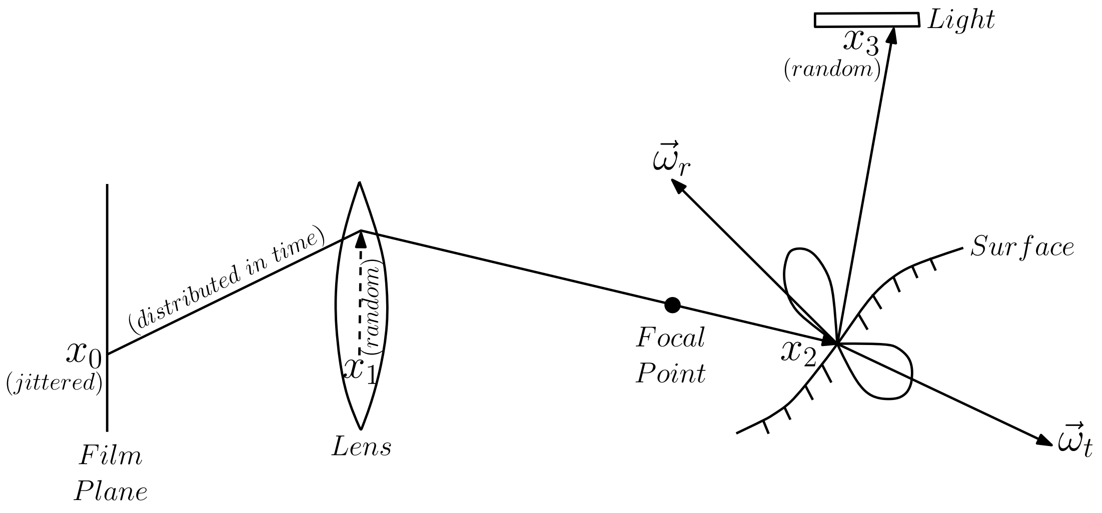


Figure 2.11*: Sampling in distributed ray tracing (amended from Cook et al. 1984, p. 5).*

However, recursive ray tracing is not a solution to the rendering equation. Firstly, the higher-order rays are traced only in specularly reflected or transmitted directions. The indirect illumination arising from other types of scattering, like translucency or gloss, is simply omitted. Furthermore, the purely specular sampling precludes the simulation of soft shadows, depth of field and motion blur. In terms of Heckbert's notation (1990), the recursive ray tracing algorithm is limited to generating paths of the form $LD?S^*E$.

Distributed ray tracing (Cook et al. 1984) overcomes the limitations of recursive ray tracing by sampling rays according to the BSDF, rather than assuming δ directions for the reflected and transmitted rays. The random sampling of the BSDF enables the simulation of fuzzy phenomena, like translucency and gloss. Moreover, effects like depth of field, motion blur and penumbrae are generated by sampling lenses, time and light sources. However, distributed ray tracing focuses most of its effort on generating higher-order rays, which contribute less to variance than primary rays (Kajiya 1986). Figure 2.11 illustrates the concepts of distributed ray tracing for a single primary ray.

The random sampling ideas, which underlie distributed ray tracing, inspired the first Monte Carlo ray tracing algorithm. Kajiya (1986) defines the light transport problem as an integral that can be solved by means of randomly sampled light transport paths.

2.5.4 Monte Carlo ray tracing

The goal of a Monte Carlo ray tracing method is to estimate the measurement equation:

$$I_j = \int_{\mathfrak{X}} f_j(\bar{x}) d\mu(\bar{x})$$

Since the publication of path tracing (Kajiya 1986), plural algorithms use *Monte Carlo integration* to solve the rendering equation. The approach is to randomly sample light transport paths, based on a probability density function p, and compute the estimator:

$$I_j \approx F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{X}_i)}{p(\bar{X}_i)} \tag{2.54}$$

where N is the number of sampled paths and $\bar{X}_i$ is a path sampled according to the probability density $p(\bar{X}_i)$. Equation (2.54) represents an unbiased estimation of the measurement I_j. This means that on average, the estimator F_N yields the correct result:

$$\mathcal{E}[F_N] = \mathcal{E}\left[\frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{X}_i)}{p(\bar{X}_i)}\right] = \frac{1}{N} \sum_{i=1}^{N} \int_{\mathfrak{X}} \frac{f_j(\bar{x})}{p(\bar{x})} p(\bar{x}) d\mu(\bar{x}) = \int_{\mathfrak{X}} f_j(\bar{x}) d\mu(\bar{x}) = I_j \tag{2.55}$$

where $\mathcal{E}[F_N]$ is the expected value of the estimator. The *expected value* or *expectation* of a random variable $Y = f(X)$ is defined as $\mathcal{E}[Y] = \mathcal{E}[f(X)] = \int_{\mathcal{D}_x} f(x)p(x)d\mu(x)$. The *variance* of the random variable Y is $\mathcal{V}[Y] = \mathcal{E}[(Y - \mathcal{E}[Y])^2] = \mathcal{E}[Y^2] - \mathcal{E}^2[Y]$.

The estimator F_N is *unbiased* if the expected value of the *error* $F_N - I_j$ is equal to zero:

$$\beta[F_N] = \mathcal{E}\left[F_N - I_j\right] = \mathcal{E}[F_N] - I_j = 0 \tag{2.56}$$

A Monte Carlo estimator, with bias $\beta \neq 0$, is *consistent* only if (Veach 1998, p. 43):

$$\lim_{N\to\infty} \beta[F_N] = \lim_{N\to\infty} \mathcal{V}[F_N] = 0 \tag{2.57}$$

To determine the accuracy of an estimator, the *mean squared error* can be computed:

$$\begin{aligned} MSE[F_N] &= \mathcal{E}\left[\left(F_N - I_j\right)^2\right] \\ &= \mathcal{E}[F_N^2] - 2\mathcal{E}\left[F_N I_j\right] + \mathcal{E}[I_j^2] = \mathcal{E}[F_N^2] - 2\mathcal{E}[F_N]I_j + I_j^2 \\ &= (\mathcal{E}[F_N^2] - \mathcal{E}^2[F_N]) + (\mathcal{E}^2[F_N] - 2\mathcal{E}[F_N]I_j + I_j^2) \\ &= \mathcal{V}[F_N] + \left(\mathcal{E}[F_N] - I_j\right)^2 = \mathcal{V}[F_N] + \beta^2[F_N] \end{aligned} \tag{2.58}$$

Equation (2.58) underlines a major difference between an unbiased and a consistent estimator. For an unbiased estimator, the error can be determined by simply evaluating its variance, i.e. $MSE[F_N] = \mathcal{V}[F_N]$, since $\beta[F_N] = 0$. Moreover, the variance of an unbiased estimator decreases linearly with the number of samples (Veach 1998, p. 39):

$$\begin{aligned} \mathcal{V}[F_N] &= \mathcal{V}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f_j(\bar{X}_i)}{p(\bar{X}_i)}\right] = \frac{1}{N^2}\mathcal{V}\left[\sum_{i=1}^{N}\frac{f_j(\bar{X}_i)}{p(\bar{X}_i)}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathcal{V}\left[\frac{f_j(\bar{X}_i)}{p(\bar{X}_i)}\right] \\ &= \frac{1}{N}\left(\int_{\mathfrak{X}}\frac{f_j^2(\bar{x})}{p(\bar{x})}d\mu(\bar{x}) - I_j^2\right) = \frac{1}{N}\mathcal{V}\left[\frac{f_j(\bar{X})}{p(\bar{X})}\right] = \frac{1}{N}\sigma^2\left[\frac{f_j(\bar{X})}{p(\bar{X})}\right] \end{aligned} \tag{2.59}$$

where $\bar{X}_i$ are independent random paths and the property $\mathcal{V}[\sigma Y_i] = \sigma^2\mathcal{V}[Y_i]$ holds for any random variable $Y_i = f_j(\bar{X}_i)/p(\bar{X}_i)$. Expression $\mathcal{V}[F_1] = \mathcal{V}\left[f_j(\bar{X})/p(\bar{X})\right]$ refers to an estimator defined on a single random path $\bar{X}$ and $\sigma[F_1] = \sqrt{\mathcal{V}[F_1]}$ is the standard deviation of the estimator F_1. Equation (2.59) is the mathematical argument behind the commonly known fact that Monte Carlo algorithms suffer from variance. Their convergence rate is $O(1/\sqrt{N})$, given the *root mean squared error* is $\sigma[F_N] = \frac{1}{\sqrt{N}}\sigma[F_1]$.

The error of a consistent estimator cannot be determined as easily, because bias is not only a function of the number of samples, but also depends on I_j (Veach 1998, p. 44).

Without bias, the *efficiency* of an estimator can be measured relative to the running time that is used to sample and evaluate the light transport paths (Veach 1998, p. 45):

$$\epsilon[F_N] = \frac{1}{\mathcal{V}[F_N]\mathcal{t}[F_N]} \tag{2.60}$$

To use Monte Carlo integration, a method must evaluate the measurement contribution function f_j (equation 2.44) and the probability density function p. The probability density function p is defined relative to the differential area-product measure $d\mu$ and depends on the random generation of the point samples that compose the given path.

A random path $\bar{x} = x_0 \dots x_k$ has a probability density equal to (Veach 1998, p. 227):

$$p(\bar{x}) = \frac{dP(\bar{x})}{d\mu(\bar{x})} = \frac{dP(x_0 \dots x_k)}{d\mu(x_0 \dots x_k)} = \frac{dP(x_0) \dots dP(x_k)}{dA(x_0) \dots dA(x_k)} = \prod_{i=0}^{k} \frac{dP(x_i)}{dA(x_i)} \qquad (2.61)$$

where $dP(x_i)/dA(x_i)$ is the probability per unit area with which vertex x_i was sampled.

The vertex of a path can be sampled either from a distribution over the surface area, or by randomly casting a ray. In the first case, $dP(x_i)/dA(x_i)$ can be used as such. When a vertex x_i is sampled by determining the intersection of a random ray with the scene geometry, the probability of the random ray must be converted to a probability measured with respect to the surface area. Let ray $r(x_{i-1}, x_{i-1} \to x_i)$ be sampled from a probability density $p^{\perp}(x_{i-1} \to x_i)$ measured with respect to the projected solid angle. The projected solid angle probability density and the area homologue are related via:

$$\frac{dP(x_i)}{dA(x_i)} = \frac{dP(x_i)}{d\vec{\omega}_o^{\perp}} \cdot \frac{d\vec{\omega}_o^{\perp}}{dA(x_i)} \quad \Leftrightarrow \quad p(x_i) = p^{\perp}(\vec{\omega}_o) \frac{d\vec{\omega}_o^{\perp}}{dA(x_i)}$$

$$\overset{(2.31)}{\Longleftrightarrow} \quad p(x_i) = p^{\perp}(\vec{\omega}_o) G(x_{i-1} \leftrightarrow x_i) \qquad (2.62)$$

where $\vec{\omega}_o = \widehat{x_i - x_{i-1}}$. Should $\vec{\omega}_o$ be sampled from a probability density measured with respect to the solid angle then, by equation (2.11), the area probability density becomes:

$$p(x_i) = \frac{p(\vec{\omega}_o)}{\left|\vec{N}_{x_{i-1}} \cdot \vec{\omega}_o\right|} G(x_{i-1} \leftrightarrow x_i) = p(\vec{\omega}_o) \frac{\left|\vec{N}_{x_i} \cdot -\vec{\omega}_o\right|}{\|x_{i-1} - x_i\|^2} \qquad (2.63)$$

where equality $p^{\perp}(\vec{\omega}_o) = p(\vec{\omega}_o)(d\vec{\omega}_o / d\vec{\omega}_o^{\perp}) = p(\vec{\omega}_o)\left(1/\left|\vec{N}_x \cdot \vec{\omega}_o\right|\right)$ holds $\forall x \in \mathcal{M}$.

Using either equation (2.62) or (2.63), the probability density $p(\bar{x})$ can be computed as the product of all the probability densities with which the vertices of the path $\bar{x}$ were sampled. Every path $\bar{x}$ is sampled by a particular strategy, with a probability density $p(\bar{x})$. Consequently, the estimator (2.54) subsumes different path sampling strategies.

2.5.4.1 Path tracing

Path tracing was the first unbiased Monte Carlo ray tracing algorithm used in computer graphics. Introduced together with the rendering equation (Kajiya 1986), path tracing generates paths that begin at the camera and are of the form $L(S|D)^*E$. It extends the ideas of distributed ray tracing, by sampling locally defined probability distributions.

Since a path starts at the camera, its first vertex is either sampled anywhere on the lens or is determined by tracing a ray, from a random position in a pixel, towards the lens. The ray that connects the image plane to the lens also determines the first intersection with the scene. The ensuing vertices are generated by sampling directions according to the BSDFs and computing the closest scene intersections. The last vertex is sampled directly on the light source. A path terminates if it randomly intersects a light source.

Unlike distributed ray tracing, path tracing does not generate multiple rays per point sample. At each point sample, the number of rays generated by distributed ray tracing is proportional to the amount of light scattered towards the camera. Instead of focusing its computational effort on a ray tree, path tracing samples just one ray per impact point. Since the branching factor is always 1, path tracing assigns the same workload to both primary and higher-order rays, with the result that variance reduces as compared to distributed ray tracing. Performance increases due to less rays traced per point sample.

Mathematically, path tracing is a Markov chain estimation of the Neumann series (equation 2.28). A stationary, discrete *Markov chain* is a sequence of random variables $\bar{x} = x_0 \dots x_k$, with an *initial probability density* $p(x_0)$ and k *transition probability densities* $p(x_{i-1} \rightarrow x_i), 1 \leq i \leq k$. Some Markov chain states are absorbing, because they lack an exitant transition. As a path, a Markov chain can be evaluated via Monte Carlo integration, with its probability being computed through equations (2.61) - (2.63).

The interesting problem that arises in light transport, is the estimation of an arbitrarily long series (equation 2.43) with a finite amount of terms. Monte Carlo integration randomly samples the estimand f_j and weights the computed values using the sampling probabilities. However, the question really posed is how to stop sampling after a finite number of terms without introducing bias. The Monte Carlo answer is *Russian roulette.*

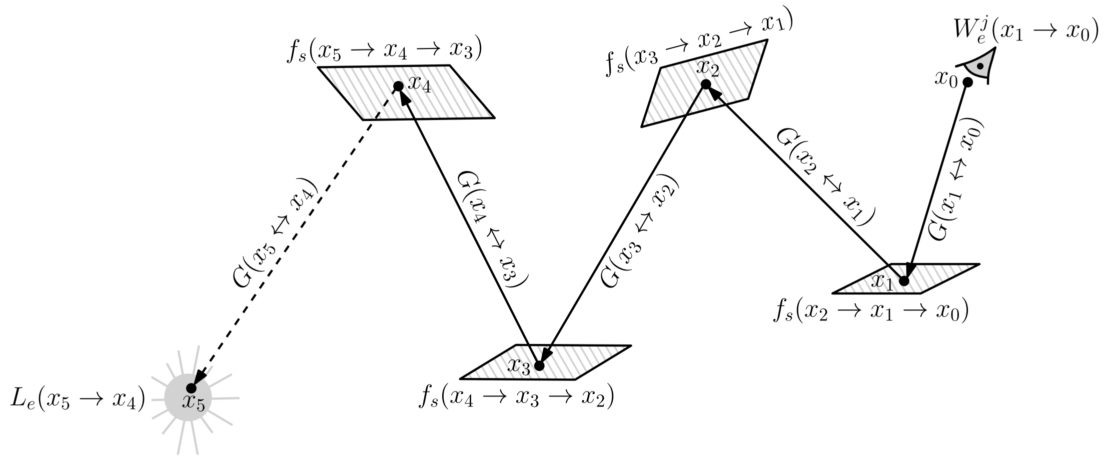
Russian roulette addresses the problem of evaluating samples whose contributions are relatively small. Given an estimator composed of several contributions $F = \sum_{i=1}^{N} F_i$, Russian roulette decides whether to assess F_i via the ensuing test (Veach 1998, p. 67):

$$F_i^* = \begin{cases} \frac{1}{q_i} F_i & \xi < q_i \\ 0 & otherwise \end{cases} \tag{2.64}$$

where F_i represents the i^{th} contribution subsumed in estimator F, ξ is a random number and q_i is the probability with which F_i is evaluated. The evaluation probability q_i is chosen based on an estimate of F_i. The estimator F_i^* depends on the evaluation probability q_i and it is unbiased whenever F_i is: $\mathcal{E}[F_i^*] = \underbrace{q_i \cdot \left(\frac{1}{q_i}\right) \mathcal{E}[F_i] + (1 - q_i) \cdot 0}_{\mathcal{E}[F_i]}$.

Russian roulette can be used to terminate infinite random walks without introducing bias. Given a random path $\bar{x} = x_0 \ldots x_i$, Russian roulette decides whether $\bar{x}$ should be extended or terminated. The decision relies on the contribution that would be obtained by further extending $\bar{x}$. A similar test to (2.64) is used with the continuation probability q_i chosen according to the amount of light that is scattered from x_i to x_{i+1}. Veach (1998, p. 310) concretely defines q_i. Though the meaning of q_i depends on the test type, q_i will be hereafter used as a continuation probability. Russian roulette can also be used to reduce visibility tests (Veach 1998, p. 317). Russian roulette increases variance, yet it can also increase efficiency (2.60) by reducing the running time (Veach 1998, p. 67).

Path tracing can use Russian roulette to unbiasedly terminate the sampling of the vertices that are not located on the lens or on a light source. The use of Russian roulette causes the probabilities densities (2.62) and (2.63) to become a product between the probability density used to sample a new vertex and the continuation probability q_i.



$f_s(x_5 \to x_4 \to x_3)$
x_4
$f_s(x_3 \to x_2 \to x_1)$
x_2
$W_e^j(x_1 \to x_0)$
x_0
$G(x_5 \leftrightarrow x_4)$
$G(x_4 \leftrightarrow x_3)$
$G(x_3 \leftrightarrow x_2)$
$G(x_2 \leftrightarrow x_1)$
$G(x_1 \leftrightarrow x_0)$
x_1
$f_s(x_2 \to x_1 \to x_0)$
x_3
$L_e(x_5 \to x_4)$
x_5
$f_s(x_4 \to x_3 \to x_2)$

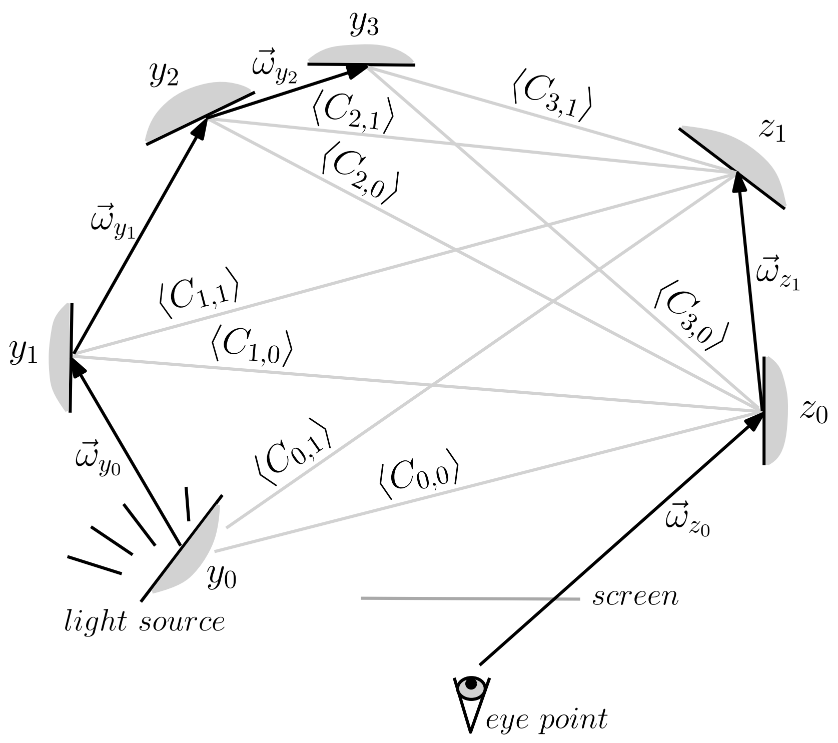
***Figure 2.12**: Sampling a unidirectional path with connection to a random light vertex.*

The measurement contribution function f_j has meaning only if the last path vertex is sampled on a light source. When a path randomly intersects a light source, it terminates naturally and its contribution can be evaluated through equation (2.54). Considering only paths that randomly intersect the light sources would prove inefficient, as paths have a low probability of randomly intersecting the light sources. In such cases, the solution would converge extremely slowly and hence would suffer from high variance.

Instead, path tracing algorithms randomly select a point on a light source and connect the given path to the sampled light vertex. Both Veach (1998, p. 311) and Pharr and Humphreys (2004, p. 723-728) discuss strategies for sampling vertices on the light sources. With their light vertices in place, paths can be evaluated via estimator (2.54). Jensen (2001, p. 43) provides a high-level description of the path tracing algorithm.

Compared to distributed ray tracing, path tracing can utilize its performance gain to generate multiple paths per pixel and thus further reduce variance. However, variance reduction is a difficult task for path tracing and generally for Monte Carlo ray tracing. Kajiya (1986) introduces five variance reduction techniques, whereas Veach (1998) provides a detailed treatment of the four predominant variance reduction categories.

Path tracing samples paths unidirectionally. Figure 2.12 depicts the sampling of a unidirectional path, which is randomly terminated at x_4 and is connected to the random light vertex x_5. The adjoint BSDF is used to evaluate the throughput at each vertex.



y_3
y_2
$\vec{\omega}_{y_2}$
$\langle C_{3,1} \rangle$
$\langle C_{2,1} \rangle$
z_1
$\langle C_{2,0} \rangle$
$\vec{\omega}_{y_1}$
$\vec{\omega}_{z_1}$
$\langle C_{1,1} \rangle$
$\langle C_{3,0} \rangle$
y_1
$\langle C_{1,0} \rangle$
z_0
$\vec{\omega}_{y_0}$
$\langle C_{0,1} \rangle$
$\langle C_{0,0} \rangle$
$\vec{\omega}_{z_0}$
y_0
light source
screen
eye point

***Figure 2.13**: The formulation of bidirectional path tracing as a connectivity strategy between light & eye subpath vertices (amended from Lafortune and Willems 1993, p. 5).*

2.5.4.2 Bidirectional path tracing

Bidirectional path tracing (Lafortune and Willems 1993; Veach and Guibas 1994) generalizes path tracing by sampling paths from both the camera and the light sources.

Veach and Guibas (1994; 1995) formulate bidirectional path tracing as a family of path sampling techniques. Each technique samples a path with a specific probability density. The path integral framework supports both the computation of the probability densities and the combination of all the probability densities with which a path could have been sampled by the distinct techniques. Known as multiple importance sampling (Veach and Guibas 1995), the latter operation serves to create low-variance estimators.

Lafortune and Willems (1993) describe bidirectional path tracing as a method that uses shadow rays to connect the vertices of a light subpath to the vertices of an eye subpath. With this approach, the generation of paths is fixed on a specific connectivity strategy. Figure 2.13 illustrates this bidirectional path connectivity and evaluation strategy.

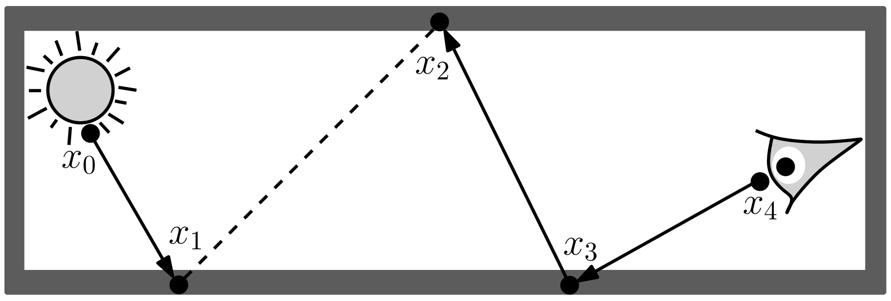


Figure 2.14*: A light transport path of length $k = 4$, created by concatenating a light subpath with 2 vertices to an eye subpath with 3 vertices (Veach 1998, p. 299).*

Bidirectional path tracing can construct paths by connecting a subpath sampled from the light source to a subpath sampled from the camera. Still, paths can be constructed generically and the connectivity strategy between the light and eye vertices can vary. For example, each eye subpath can be connected to a randomly selected light subpath. Conversely, a subset of eye subpaths can be connected to the same light subpath. Such connectivity strategies can be regarded as optimizations for efficient path generation.

Another difference between the two versions of bidirectional path tracing, refers to the set of estimators. Lafortune and Willems (1993) do not account for estimators defined on paths with zero or one eye vertices and with zero light vertices. Thus, the effects associated with the missing estimators are not simulated. For example, the absence of naïve path tracing estimators precludes effects like caustics and directly visible lights. Other differences regard direct lighting strategies, non-local sampling strategies, image independent solutions and variance reduction techniques (Veach 1998, p. 326). Hence, the ensuing discussion focuses on the approach proposed by Veach and Guibas (1994).

A path generated with bidirectional path tracing consists of a light and an eye subpath. Both subpaths are stationary, discrete Markov chains. The *light subpath* is generated from the light source with s random vertices $y_0 \dots y_{s-1}$. The *eye subpath* is generated from the camera with t random vertices $z_0 \dots z_{t-1}$. The subpath lengths are determined via Russian roulette (Veach 1998, p. 309). Vertices y_{s-1} and z_{t-1} are connected to form a light transport path $\bar{x} \equiv \bar{x}_{s,t} = y_0 \dots y_{s-1} z_{t-1} \dots z_0$. The vertices that are connected are called *connecting vertices* and the edge between them is called the *connecting edge.* Figure 2.14 shows the bidirectional sampling of a light transport path of length $k = 4$.

The above designation is not limited to the end vertices. Veach (1998, p. 300) proposes an approach that joins each prefix of the light subpath to each suffix of the eye subpath and thus efficiently draws additional samples from the same pair of light and eye subpaths. Within this context, all the prefix-suffix pairs represent connecting vertices.

When subpaths randomly intersect the lens or a light source, they represent complete unidirectional paths and they need not be connected to other subpaths. The rest of the subpaths must be connected, even if they comprise a single vertex. Zero and one vertex subpaths require special treatment and they are discussed by Veach (1998, p. 310-314).

A complete light transport path can be sampled in more than one way, through a family of sampling techniques. These sampling techniques are obtained by varying the number of vertices in both the light and the eye subpath. For a path of length $k = s + t - 1$ there are $k + 2$ different sampling techniques that can be used to generate the path. Each of the $k + 2$ sampling techniques corresponds to a different probability density over the space of paths. This means that each technique samples different factors from the measurement contribution function f_j (equation 2.44) and thus accounts for distinct illumination effects. Bidirectional path tracing takes advantage of all these nuances and combines all the $k + 2$ sampling techniques in the following multi-sample estimator:

$$F = \sum_{s \geq 0} \sum_{t \geq 0} w_{s,t}(\bar{x}_{s,t}) \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} \tag{2.65}$$

where $w_{s,t}$ is the weighting function associated with a combination strategy, like the power heuristic (Veach and Guibas 1995). Chapter 3, subsection 3.1.1, details the local path sampling techniques. Chapter 3, subsection 3.1.2, delineates the computation of the above multi-sample estimator. The latter subsection provides complete derivations for the measurement contribution function, probability density and weighting function.

A complete light transport path cannot be constructed, if the connecting vertices are obstructed. Likewise, two subpaths cannot be connected if the BSDF of one connecting vertex does not scatter light towards the other vertex. For example, the connection attempt fails whenever a connecting vertex has a purely specular BSDF. The rationale is that the solid angle around the specular direction is zero and so is the probability for the connecting edge to be established. In fact, bidirectional path tracing can only connect subpaths that form DD edges, simply because there is a very low probability for a specular direction to coincide with the connecting edge. Generally, bidirectional path tracing cannot handle Dirac delta functions, like point or directional light sources, pinhole cameras and perfectly specular surfaces. Path vertices from δ functions have zero probability of being sampled by other techniques than the one that generated them.

There are various remedies to such difficulties. The simplest, is to restrict the use of Dirac delta components and thus benefit from unbiased results. The second solution is to allow Dirac delta components at the cost of various estimators. For example, point light sources could be admitted to the detriment of estimators defined on light subpaths with zero or one vertices. Veach (1998) proposes non-local path sampling approaches.

Bidirectional path tracing also struggles with environments that represent the outdoors or portals, such as light seeping at the edges of a window covered by an opaque curtain. However, bidirectional path tracing is a state-of-the-art algorithm that can robustly simulate diverse illumination effects for many physically valid scene configurations. Jensen (2001, p. 47) shows a schematic flow of the bidirectional path tracing algorithm.

2.5.4.3 Metropolis light transport

Metropolis light transport (Veach and Guibas 1997) was introduced as a more efficient approach to sampling the path space. The fundamental idea is to generate the Markov chains over the path space, according to the contributions they bring to the ideal image.

Unlike path tracing and bidirectional path tracing, Metropolis light transport does not explore the path space by generating numerous different Markov chains. Instead, it samples the path space locally, by applying small mutations to a number of seed paths.

Consider for the moment that a single seed path $\bar{X}_0$ is generated according to a convenient probability density p_0. The goal is to generate a sequence of paths $\bar{X}_1 \ldots \bar{X}_N$, in which each path $\bar{X}_{i+1}$ is obtained by mutating its predecessor $\bar{X}_i, 0 \le i \le N-1$. The applied mutations can take any form and they usually modify a few path vertices.

Each mutation has an *acceptance probability* that is defined based on the contributions of the mutated and mutant paths. Metropolis light transport computes the acceptance probabilities, such that every path $\bar{X}_i$ is distributed proportionally to the *image contribution function* (Veach 1998, p. 332). The image contribution function $f(\bar{X}_i)$ relates the light flow along $\bar{X}_i$ to image pixels and it is similar to the measurement contribution function f_j (equation 2.44). Veach (1998, p. 338) abstracts the sensor terms in a *filter function* $\hbar_j(\bar{x})$ and redefines the measurement equation as follows:

$$I_j = \int_{\mathfrak{X}} \hbar_j(\bar{x}) f(\bar{x}) d\mu(\bar{x}) \tag{2.66}$$

where $\hbar_j(\bar{x})$ correlates the contribution of $\bar{x}$ with pixel j and the image contribution function f contains the remaining terms of the measurement contribution function f_j. The measurement I_j can be evaluated via Monte Carlo integration (estimator 2.54).

Every random path $\bar{X}_i$ is sampled with a probability density p_i (Veach 1998, p. 334):

$$p_i(\bar{y}) = \int_{\mathfrak{X}} p(\bar{x} \to \bar{y}) p_{i-1}(\bar{x}) d\mu(\bar{x}) \tag{2.67}$$

where p is the *transition function* that represents the probability density associated with mutating $\bar{X}_{i-1} = \bar{x}$ into $\bar{X}_i = \bar{y}$. The mutation of the path $\bar{X}_{i-1}$ usually involves adding, replacing, deleting or perturbing a few vertices. Veach (1998) proposes three types of mutations: bidirectional mutations, perturbations and lens subpath mutations. Figure 2.15 illustrates a bidirectional mutation together with a caustic perturbation.

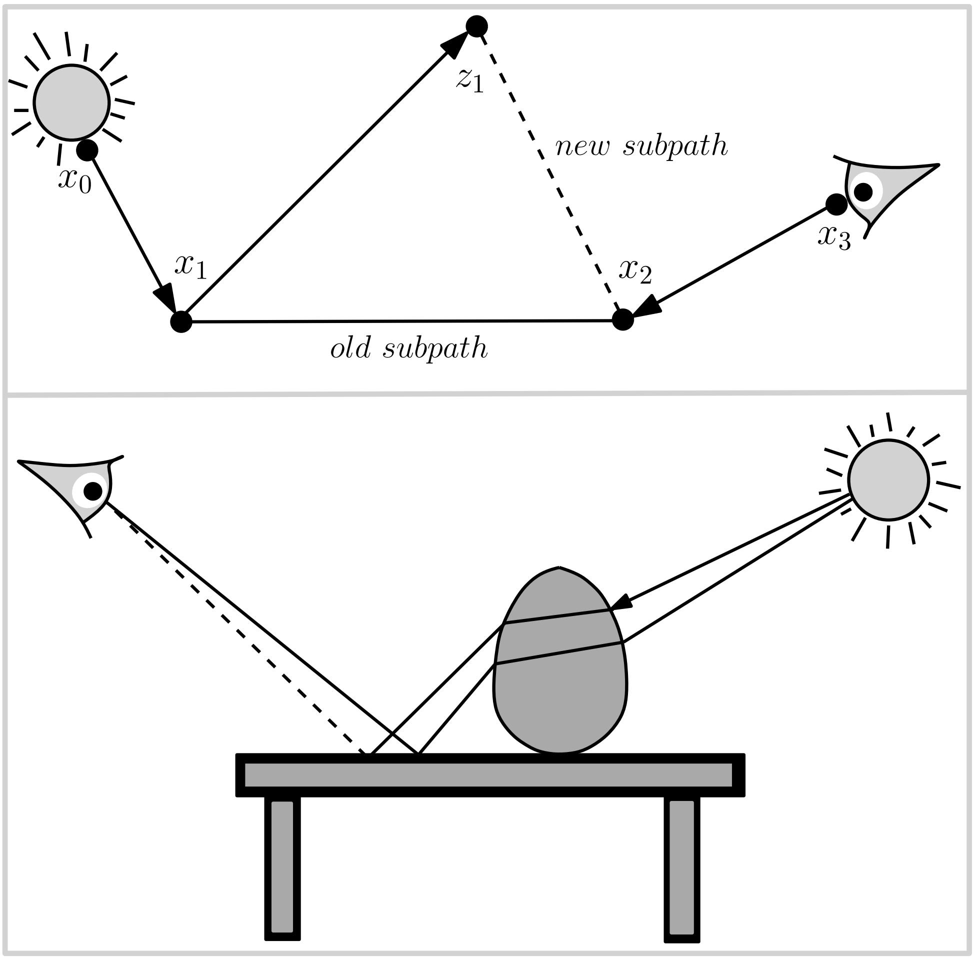


Figure 2.15*: Example of bidirectional mutation (top) & caustic perturbation (bottom). The bidirectional mutation deletes the edge x_1x_2 of the path $\bar{x} = x_0x_1x_2x_3$ and replaces it with a new vertex z_1 yielding the mutated path $\bar{y} = x_0x_1z_1x_2x_3$. The caustic perturbation generates a new path by perturbing the light source ray and tracing it through similar bounces to the initial path (amended from Veach 1998, p. 348 & 353).*

A tentative sample is accepted/rejected based on the acceptance probability $a(\bar{x} \to \bar{y})$. The acceptance probability is derived by assuming an equilibrium distribution, i.e. p_{i-1} is assumed to be proportional to f. To preserve the equilibrium, any transition must be proportional to the image contribution function and must be equipoised by its reverse:

$$f(\bar{x})p(\bar{x} \to \bar{y})a(\bar{x} \to \bar{y}) = f(\bar{y})p(\bar{y} \to \bar{x})a(\bar{y} \to \bar{x}) \qquad (2.68)$$

This condition is known as *the detailed balance*. Due to parity reasons the reversed transition is always accepted via $a(\bar{y} \to \bar{x}) = 1$. By maximizing the probability of the reversed transition, the acceptance probability can be defined as (Veach 1998, p. 336):

$$a(\bar{x} \to \bar{y}) = \min\left\{1, \frac{f(\bar{y})p(\bar{y} \to \bar{x})}{f(\bar{x})p(\bar{x} \to \bar{y})}\right\} \qquad (2.69)$$

However, the probability densities associated with $\bar{X}_i$ become proportional to f only as $N \to \infty$. The closer the mutated paths are to the seed path $\bar{X}_0$, the stronger they are influenced by it. The pervasive effect of the seed path causes the solution to suffer from *start-up bias*. Veach (1998, p. 339) suggests a method that eliminates the start-up bias.

So far, a single path was considered for the generation of the mutated paths. In practice, it is convenient to sample multiple initial paths and seed the Metropolis light transport algorithm with each of them. The seeds can be generated with other unbiased path sampling algorithms, like bidirectional path tracing. Jensen (2001, p. 49) provides a high-level description of the Metropolis light transport algorithm, using multiple seeds.

Metropolis light transport is a sophisticated algorithm that can simulate difficult light configurations, such as portals concentrating most of the scene illumination. Because it samples the path space locally, Metropolis light transport can detect and efficiently explore regions with significant illumination. However, it is quite sensitive to the type of mutations. For example, restrictive mutations may cause the algorithm to get caught in some region of the path space, with the result that the solution converges slower. Mutations and their associated probabilities may also lead to correlation and variance.

Metropolis light transport concludes the section on the versatile and robust class of Monte Carlo ray tracing algorithms. These algorithms can simulate a variety of light effects for generic scene configurations. Hence, Monte Carlo ray tracing fulfils the prerequisite of the light transport framework and was identified as a development basis.

2.5.5 Photon mapping

Photon mapping (Jensen 2001) is a two-pass global illumination algorithm, aiming to create images of generic environments free of high-frequency noise. Founded on the idea that many environments contain regions over which illumination varies smoothly, photon mapping computes light information at various scene locations and stores it in a structure that can be used afterwards to render the environment. The first phase of the photon mapping algorithm is *photon tracing*, whereas the second phase is *rendering*.

In the photon tracing phase, the photons emitted from the light sources are traced several bounces until they are absorbed. At diffuse impact points, the photon and its information (e.g. position, incoming direction, power, etc.) are stored in the *photon map*. The photon map is a global cache which enables the estimation of the radiance at various surface locations. The storing of photons only at diffuse surfaces, is motivated by the low probability that a photon has to arrive from a specular direction. Specular effects are generated via ray tracing. Jensen (2001) discusses the *emission of photons* from various light sources. The *scattering of photons* is decided stochastically based on the scattering properties of the surfaces. Russian roulette (Jensen 2001, p. 62-63) is used to decide whether a photon is reflected, transmitted or absorbed. A key difference between rays and photons, is that rays carry radiance and photons carry power. The used entity may cause different interactions (e.g. photons do not refract like rays do).

In the rendering phase, the photon map is used to estimate the radiance at different surface locations. The radiance estimate is computed using *kernel density estimation*. For each point sample, the kernel density estimation determines the N nearest photons located within a given distance (Jensen 2001, p. 72). Several volumes can be used to locate the photons. The chosen volume determines the speed and the accuracy with which the nearest photons are identified. For example, the sphere has the advantage that the distance and the projected area can be easily computed. However, volumes like the ellipsoid may have less false positives. For the sphere the radiance estimate is:

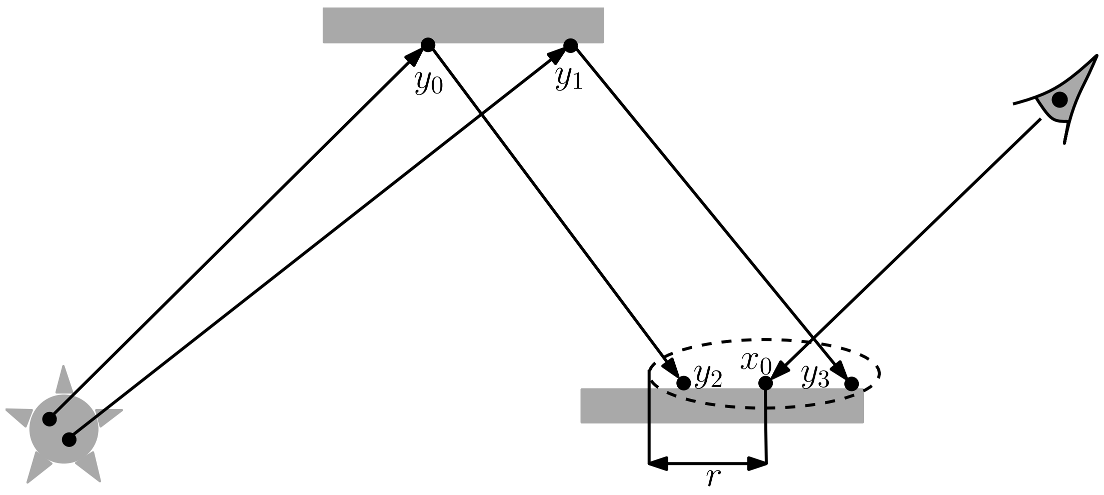


Figure 2.16*: Two nearest photons y_2 and y_3 are identified around point sample x_0.*

$$L_o(x, \vec{\omega}_o) = \frac{1}{\pi r^2} \sum_{i=1}^{N} f_s(x, \vec{\omega}_i, \vec{\omega}_o) \Delta\Phi(x, \vec{\omega}_i) \tag{2.70}$$

where r is the sphere radius within which the nearest photons lie. The surface around point x is assumed to be flat. Figure 2.16 shows the identification of the nearest photons.

The radiance estimate is susceptible to both the size of the photon map and the number of nearest neighbours. A small photon map tends to blur the edges of sharp illumination features. By weighting photons based on their distance from the point sample, 2D filters can enhance edge features. However, corners and edges may introduce errors in the estimate if the size of the photon map and number of the nearest photons are too small.

Jensen (2001) proposes two approaches to using the photon map. The first approach uses kernel density estimation to render diffuse scattering and recursive ray tracing to render specular effects. The second approach uses distributed ray tracing to render the entire image. The radiance computation is effectively a combination of accurate and approximate evaluations. The accurate evaluation is used for directly viewed surfaces, specular phenomena and ray lengths smaller than a given threshold. The approximate evaluation is used for diffuse scattering and primary rays with small contributions. Depending on requirements, the photon map and the radiance estimate can be used for both accurate and approximate evaluations. For example, they can be utilized to accurately compute caustics as well as to approximate direct illumination.

Unlike Monte Carlo ray tracing, photon mapping is biased. The source of bias is the kernel density estimation, which acts as a low-pass filter. Though the kernel density estimation reduces variance, it introduces multifold bias (Herzog et al. 2007). The *boundary bias* is caused by the fewer observations that can be averaged at the boundary of the evaluated region. *Topological bias* arises from the assumption that surfaces are flat around the point samples. *Proximity bias* results from setting a volume radius and influencing the correctness of the photon density used to estimate radiance.

However, photon mapping is consistent (equation 2.57). *Progressive photon mapping* (Hachisuka et al. 2008; Hachisuka and Jensen 2010) uses a single ray tracing pass and multiple photon tracing passes to compute and refine an estimate that converges to the correct solution as photons are added. The *ray tracing pass* determines all the visible surfaces by tracing paths through every pixel. Each path vertex, located on a surface with a diffuse component in the BSDF, is stored together with pertinent information, such as position, normal, incident direction, power, photon count, radius, etc. Every *photon pass* computes a photon map, stores the contribution of the nearest photons at the apposite point samples and then discards the photon map, so that memory can be reused in the next passes. Each pass reduces the volume radius and the proximity bias.

Herzog et al. (2007) reduce the geometric biases. However, the proposed photon ray splatting technique does not provide a complete global illumination solution. Rather, it defines a new density estimation method, which processes complex geometric shapes and reduces the low frequency noise specific to photon mapping. Ritschel et al. (2012) analyse other interactive methods, which either improve the quality of the illumination effects or reduce the execution costs of certain algorithmic passes, like final gathering.

Among these, image space photon mapping (McGuire and Luebke 2009) accelerates two of the most expensive steps of photon mapping. Firstly, the proposed algorithm rasterizes a bounce map from the viewpoint of each light source, storing for each pixel the closest intersected surface point, the power of the photon incident at that point and the scattering direction of the photon. The bounce map serves only as an intermediate step in interrupting and moving the photon tracing process from the GPU to the CPU.

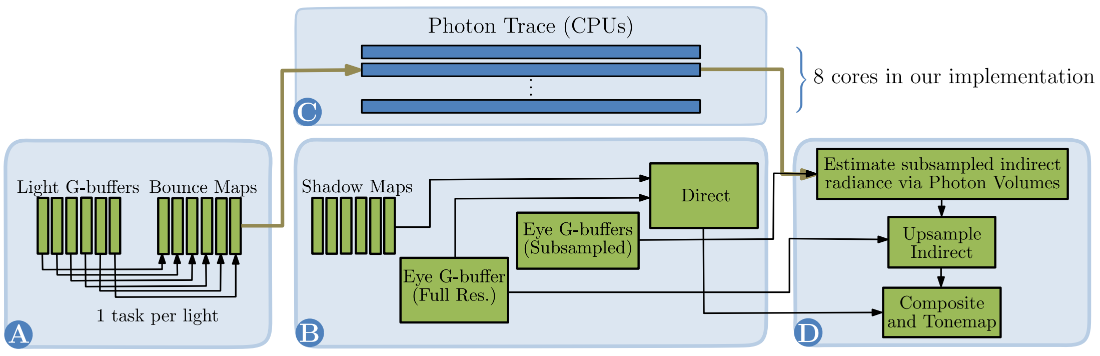


***Figure 2.17**: Image space photon mapping pipeline. The thick arrows synchronize CPU step C with GPU steps A & D (amended from McGuire and Luebke 2009, p. 4).*

After tracing the photons and constructing the photon map, the proposed algorithm estimates radiance by scattering photons to nearby pixels. For each photon a volume is rasterized and illumination is computed for all the visible surfaces from within the volume, by constructing the density estimate from the parameters of the eye geometry buffer (BSDF parameters, sample position, surface normal and depth value). Splatting photon volumes and shading all the pixels for which the density estimation kernel has significant value, effectively eliminates the costly gathering step and renders dynamic scenes with 2-megapixels resolution and interactive framerates. However, image space photo mapping works only with point emitters and pinhole cameras that have unique projection centres. Moreover, purely reflective and transmissive surfaces are rendered separately via ray tracing or rasterization approximations. Loss of indirect illumination is also observed when the camera is allowed to closely view a surface, due to clipping photon volumes before clipping the surface. To avoid reaching the GPU fill-rate and attain high performance, the proposed algorithm performs an upsampling step which introduces bias by interpolating low-resolution radiance with the high-resolution eye geometry buffer. Figure 2.17 illustrates the image space photon mapping pipeline.

Progressive photon relaxation (Spencer and Jones 2013) improves the simulation of caustic illumination by progressively removing noise and minimizing residual bias. To do so, the algorithm executes one pass of persistent photon tracing and multiple passes of transient photon tracing. The persistent photons are stored in the photon map and their locations are correlated with a flux density function via Voronoi tessellation.

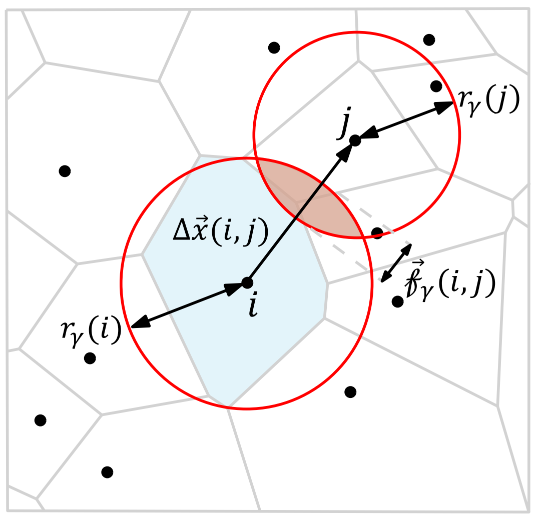


***Figure 2.18**: The relaxation operation repels photon i from its neighbour j with a force f_γ such that its bounding sphere grazes that of j. The relaxation force f_γ is computed using radii $r_\gamma(i)$ and $r_\gamma(j)$. Both radii are determined by adjusting the radii estimated from the photon distribution based on the ratio between the mean transient flux and the flux density over each cell (amended from Spencer and Jones 2013, p. 6).*

The transient photons serve to progressively refine the radiance estimate and are destroyed once their contributions have been accumulated to the flux of the photons whose Voronoi cells they intersected. The transient photon contributions are also used during a relaxation step, to compute the force which repels/attracts adjacent persistent photons. The photon relaxation is intended to create a blue noise distribution by manipulating the photon Voronoi cells such that their areas can be approximated relatively well by the density estimation kernel. The migration of persistent photons is constrained by adequately attenuating the relaxation force. The blue noise distribution locally minimizes the variance in cell areas and nearly-homogenizes the photon flux. After sufficient iterations the noise due to the photon distribution is diffused. The advantages of such an approach are low-bandwidth kernels, constant memory bound, noise removal, view independence and vectorization suitability. Yet, the proposed algorithm suffers from bias that cannot be completely eliminated since the relaxation step uses a density estimation kernel which may misestimate the Voronoi cell areas and prevent them from appropriately estimating the flux density. For materials other than perfectly diffuse ones, the approach also fails to completely eliminate noise due to the sparse number of photons that approximate the incident illumination. Figure 2.18 exemplifies the repulsion of two neighbouring photons via the relaxation force.

Unlike Monte Carlo ray tracing, photon mapping excels at simulating S^+DS^+ paths. Such paths regard caustics specularly reflected or transmitted towards the eye and are difficult to simulate via Monte Carlo ray tracing, because specular directions entail a zero solid angle and a very low probability of being sampled by other techniques than the ones that generated them. This efficiency of simulating illumination effects with reduced noise, caused photon mapping to be identified as a second potential basis for the development of the light transport framework. Jensen (2001) provides both the pseudocode for various steps and the implementation of the photon mapping algorithm.

2.5.6 Unified estimation

Unified estimation (Georgiev et al. 2012; Hachisuka et al. 2012) introduces a common theoretical framework for bidirectional path tracing and photon mapping. The incompatibility between the two algorithms stemmed from the estimation of radiance. Bidirectional path tracing estimates the measurement contribution function by sampling and evaluating light transport paths. Photon mapping uses kernel density estimation to compute radiance at various surface locations. When expressed in the same framework, these algorithms explore spaces of different dimensionality for paths of equal length. Vertex connection and merging (Georgiev et al. 2012) and unified path sampling (Hachisuka et al. 2012) effectively combine the two algorithms.

The unified estimation algorithms address the difficulty of Monte Carlo ray tracing in simulating S^+DS^+ light transport. The core idea is to adapt photon mapping to the path integral framework and combine it with bidirectional path tracing through multiple importance sampling (Veach and Guibas 1995). The resultant solutions comprise an enhanced set of sampling techniques, which can simulate difficult light transport paths.

Jensen (2001) also explores the idea of combining photon mapping with ray tracing. For instance, one proposed approach to using photon mapping relies on a caustics photon map to render caustics, on ray tracing to accurately compute direct illumination and on a global photon map to approximate direct/indirect/caustic illumination. Still, such estimation heuristics are suboptimal and they retain the formulation dichotomy.

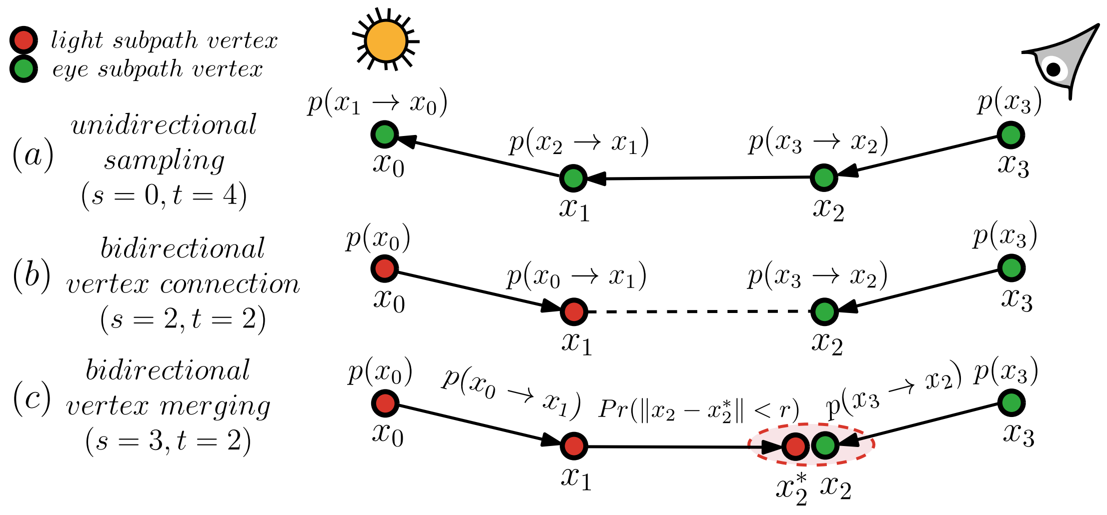

light subpath vertex
eye subpath vertex
(a) unidirectional sampling (s = 0, t = 4)
(b) bidirectional vertex connection (s = 2, t = 2)
(c) bidirectional vertex merging (s = 3, t = 2)
p(x1 → x0)
p(x2 → x1)
p(x3 → x2)
p(x3)
p(x0)
p(x0 → x1)
Pr(‖x2 − x2*‖ < r)
x0
x1
x2
x3
x2*

***Figure 2.19**: Techniques for sampling a path of length* $k = 3$ *(Georgiev et al. 2012, p. 3).*

In bidirectional path tracing, a path of length k is generated with one of the $k+2$ sampling techniques and its contribution is weighted through a heuristic that combines all of the $k+2$ techniques. However, only the unidirectional techniques in the $k+2$ family can sample S^+DS^+ paths, since the probability densities for connecting specular vertices are extremely low. Still, unidirectional sampling techniques have rather low probability densities and yield a high-variance estimator. The reformulation of photon mapping as a path sampling approach is meant to add more sampling techniques to the multi-sample estimation and facilitate the simulation of difficult light transport paths.

A path sampled with a photon mapping technique consists of a light subpath $x_0 \dots x_s^*$ and an eye subpath $x_s \dots x_k$. Vertex x_s^* represents a photon location, reached in s bounces from the light source. Vertex x_s is a point sample traced from the camera, for which the radiance estimate must be computed. Hence, a complete light transport path $\bar{x}^* = x_0 \dots x_s^* x_s \dots x_k$ has $k+2$ vertices. For the same length k, a path sampled with bidirectional path tracing has $k+1$ vertices. A path with $k+2$ vertices, has a probability density defined on a higher-dimensional differential area-product measure. Figure 2.19 depicts the difference between bidirectional and photon mapping sampling.

Vertex connection and merging solves the dimensionality conflict by removing x_s^*. To this end, the measurement contribution function $f_j(\bar{x}^*)$ is also integrated relative to x_s^*:

$$\begin{aligned}
I_j \quad = \quad & \int\limits_{\mathcal{M}^{k+1}} \left(\int\limits_{\mathcal{M}} L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \prod_{i=1}^{s-1|x_s=x_s^*} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \right. \\
& \mathcal{K}_r(\|x_s^* - x_s\|) f_s(x_s, x_{s-1} \to x_s^*, x_s \to x_{s+1}) G(x_s \leftrightarrow x_{s+1}) \\
& \left. \prod_{i=s+1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1})\, W_e^j(x_{k-1} \to x_k) dA(x_s^*) \right) d\mu(\bar{x}) \\
= \quad & \int\limits_{\mathcal{M}^{k+1}} \left(\int\limits_{\mathcal{M}} f_j(\bar{x}^*) dA(x_s^*) \right) d\mu(\bar{x}) \qquad (2.71)
\end{aligned}$$

where $\mathcal{K}_r$ is the density estimation kernel of radius r and $\int_{\mathcal{M}} f_j(\bar{x}^*) dA(x_s^*)$ corresponds to blurring by $\mathcal{K}_r$. The evaluation $f_s(x_s, x_{s-1} \to x_s^*, x_s \to x_{s+1})$ is characteristic to photon mapping, where $x_{s-1} \to x_s^*$ is the incident direction of the photon arriving at x_s^*.

Unified path sampling does not reduce the dimensionality of the extended path, instead it extends a path sampled with bidirectional path tracing via *vertex perturbation*. A path $\bar{x} = x_0 \dots x_{s-1} x_s \dots x_k$ is recreated in the extended path space by appending a new vertex x_s^* to the light subpath. The new vertex is generated by perturbing x_s with a probability density equal to the density estimation kernel: $p(x_s \to x_s^*) = \mathcal{K}_r(\|x_s^* - x_s\|)$. For a spherical volume, x_s^* is sampled on the disk that surrounds the point sample x_s. The measurement contribution function for a path $\bar{x}^* = x_0 \dots x_{s-1} x_s^* x_s \dots x_k$ is given by:

$$f_j(\bar{x}^*) = f_j(x_0 \dots x_s^*) \mathcal{K}_r(\|x_s^* - x_s\|) f_s(x_s, x_{s-1} \to x_s^*, x_s \to x_{s+1}) f_j(x_s \dots x_k) \quad (2.72)$$

The complemental approaches of the two unified estimation algorithms also regard the computation of the probability density. In vertex connection and merging, a path $\bar{x}^*$ is accepted only if the photon location x_s^* is within a distance r from the point sample x_s. Using an acceptance probability $P_{acc}(\bar{x}^*)$, the probability density of $\bar{x}^*$ is:

$$p_{VM}(\bar{x}^*) = P_{acc}(\bar{x}^*) p(x_0) \dots p(x_{s-1}) p(x_s) \dots p(x_k) = P_{acc}(\bar{x}^*) p_{VC}(\bar{x}) \qquad (2.73)$$

where $p_{VC}(\bar{x})$ is defined on a $k + 1$ dimensional differential area-product measure. To reduce the extra dimension, $P_{acc}(\bar{x}^*)$ integrates over the points within distance r of x_s:

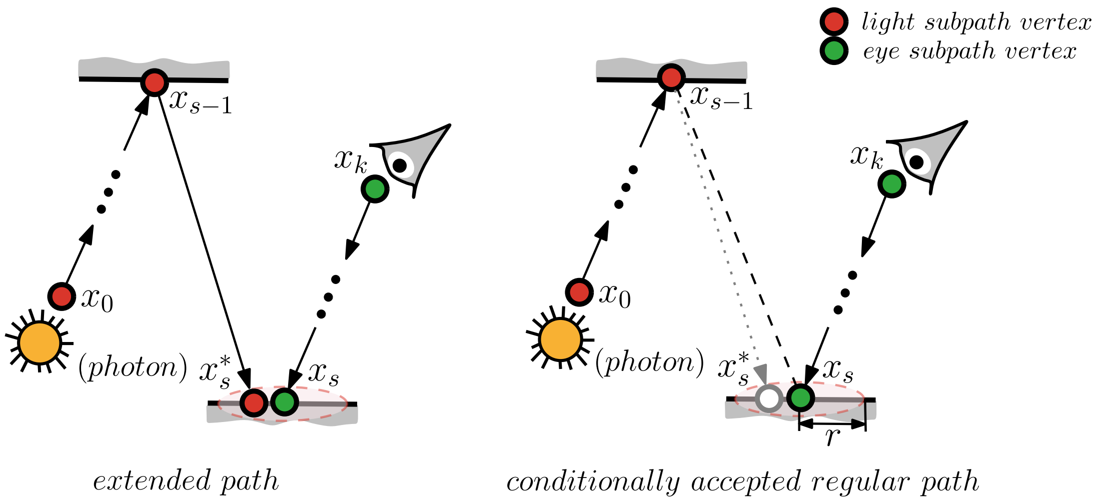


Figure 2.20: *The k-length regular path (right) is derived from the $(k+1)$-length extended path (left) iff photon x_s^* is within distance r of x_s (Georgiev et al. 2012, p. 4).*

$$P_{acc}(\bar{x}^*) = \int_{\mathcal{D}_r} p(x_{s-1} \to x)dx \approx |\mathcal{D}_r| p(x_{s-1} \to x_s^*) \approx \pi r^2 p(x_{s-1} \to x_s^*) \quad (2.74)$$

Without simplifications, the above integral is insoluble. Georgiev et al. (2012) assume that $\mathcal{D}_r = \{x \in \mathcal{M} \mid \|x_s - x\| \leq r\}$ is a disk of radius r and $p(x_{s-1} \to x)$ is constant inside it. Figure 2.20 shows the conditional sampling of a regular path from an extended one.

By extending a path sampled with bidirectional path tracing, unified path sampling can compute the probability density regardless of the approach that generated the path:

$$p(\bar{x}^*) = \big(p(x_0) \dots p(x_{s-1})\big) p(x_s^*) \big(p(x_s) \dots p(x_k)\big) \quad (2.75)$$

where $p(x_s^*)$ depends on whether x_s^* was sampled from x_{s-1} or by perturbing x_s. Thus, $p(\bar{x}^*)$ is used on paths sampled with both bidirectional path tracing and photon mapping.

Vertex connection and merging (Georgiev et al. 2012, p. 5) and unified path sampling (Hachisuka et al. 2012, p. 4) estimate the measurement equation I_j similarly, by defining a multi-sample estimator, which combines paths generated with both bidirectional and density estimation techniques. For a path of length k, the former algorithm uses $k-1$ density estimation techniques, whereas the latter algorithm uses $k+1$ such techniques. The difference in the number of density estimation techniques is the result of the path contraction or extension. Still, both algorithms generate the same range of light effects.

In being a combination of the underlying algorithms, the multi-sample estimator may suffer from both variance and bias. To reduce them, its progressive variant generates and averages multiple independent rendering iterations. The progressive estimator converges asymptotically as fast as bidirectional path tracing, but is as efficient as photon mapping at simulating S^+DS^+ light transport. It is also only consistent, just like progressive photon mapping. Caustics reflected by highly glossy surfaces are not simulated efficiently, as the basic algorithms do not have optimal sampling techniques.

Apparently, unified path sampling provides a simpler approach to adapting photon mapping to the path integral framework. Nevertheless, it entails several manoeuvres.

The generation of a new vertex is complicated in practice. The vertex x_s^* must be sampled within the support of the density estimation kernel and this operation entails the projection of a volume onto a potentially complex surface. Hachisuka et al. (2012) simplify the implementation by approximating the contribution of an extended path $\bar{x}^*$ through *virtual perturbation*. That is, the last vertex of the eye subpath x_s is set as the newly sampled vertex and the actual perturbation is not executed. Assuming a constant kernel $\mathcal{K}_r = 1/(\pi r^2)$ and a uniform distribution within its support, the authors prove that the contribution of an extended path is equivalent to that of a path sampled via bidirectional path tracing. Despite being approximate, virtual perturbations are claimed to yield a consistent estimator if proper multiple importance sampling weights are used.

By combining bidirectional path tracing and photon mapping, the unified estimation algorithms create an extended set of sampling techniques and address difficult light transport scenarios. Unified estimation was identified as a third development candidate.

2.5.7 Instant radiosity

As an alternative to radiosity, instant radiosity (Keller 1997) uses the concept of *virtual point light* (VPL) to address the discretization artefacts and the computational complexity that generally affect the radiosity algorithms. Like photon mapping, the instant radiosity algorithm can be discussed in terms of two major execution steps.

In the *first step*, photons are emitted and traced in the scene. For each light source, N particles are mapped onto its surface and traced into the scene using the first few components of low-discrepancy sequences (e.g. Halton sequence). From the first $\lfloor \bar{\rho} N \rfloor$ non-attenuated primary samples, diffuse propagation directions that follow a cosine-weighted distribution are sampled using the next components of each low-discrepancy sequence. The term $\bar{\rho}$ represents the average scene reflectance, defined as follows:

$$\bar{\rho} = \frac{\sum_{k=1}^{\mathcal{M}} \rho_{d,k} |A_k|}{\sum_{k=1}^{\mathcal{M}} |A_k|} \tag{2.76}$$

where $\mathcal{M}$ is the set of scene surfaces, A_k is the k^{th} surface area and $\rho_{d,k}$ is the diffuse reflectance (Jensen 2001, p. 21). The average reflectance also determines the absorption of particles. With each bounce i a total number of $\lfloor \bar{\rho}^i N \rfloor$ photons are further propagated.

Paths generated in this manner are known as *quasi-random walks*. The vertices of each path are used as virtual point lights in the computation of indirect illumination. The radiosity equation, used to assess the radiosity equilibrium, is defined as (Keller 1997):

$$B_j = \frac{1}{|\Psi_j|} \sum_{s=1}^{\infty} \int_{\Psi_j} \int_{\mathcal{M}^{s+1}} \int_{\mathcal{S}_e} f_j(y_0, \dots, y_s) f_r(y_{s-1} \to y_s \to z_1) G(y_s \leftrightarrow z_1) \\ f_r(y_s \to z_1 \to z_0) G(z_0 \leftrightarrow z_1) dy_0 dy_1 \dots dy_s dz_1 d\Psi \tag{2.77}$$

where Ψ_j is the j^{th} pixel of the image, $\mathcal{S}_e$ is the support of the light sources, f_j is the power contribution function, i.e. the integrand of equation (2.48), and z_1 is one of the potentially many primary samples, determined by tracing a ray through the j^{th} pixel. For purely diffuse scenes, the BRDFs in the radiosity equation can be reduced to ρ_d/π.

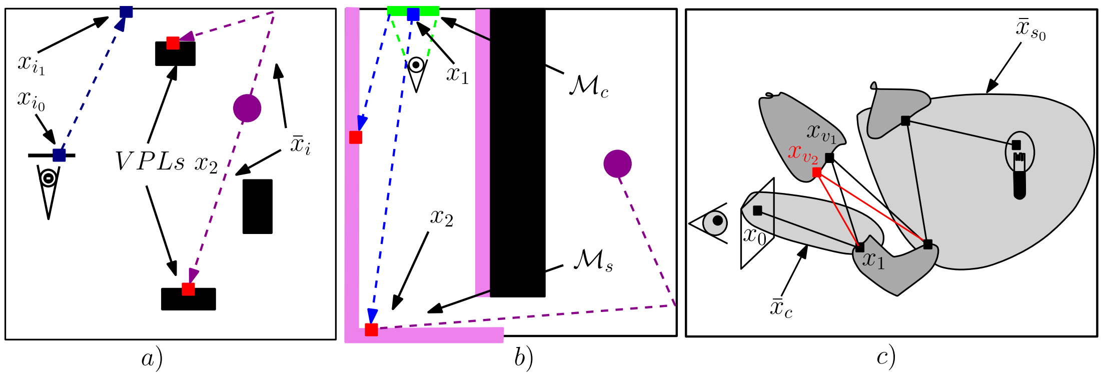


***Figure 2.21**: Bidirectional and Metropolis sampling of VPLs. Quadrangle a) displays standard instant radiosity, whereas quadrangle b) displays the sampling of reverse VPLs. The reverse VPLs are sampled from the camera via subpaths of length $k = 2$. Their radiance field is estimated by connecting them to standard VPLs sampled from the light sources (magenta lines). Quadrangle c) displays the Metropolis sampling of VPLs. The additional VPL x_{v_2} is sampled by mutating the light subpath $\bar{x}_{s_0}$, which yielded VPL x_{v_1} (amended from Segovia et al. 2006, p. 4 & Segovia et al. 2007, p. 5).*

In the *second step*, quasi-Monte Carlo integration is performed by rendering the scene with shadows for each virtual point light and summing the images in an accumulation buffer. Each image is rendered using stencil shadow volumes, i.e. by evaluating the shadow volumes of the primitives against the scene depth buffer and creating a mask in the stencil buffer. The radiance of each image is scaled by the weight $1/\lfloor\bar{\rho}^i N\rfloor$ to account for attenuation by the BRDFs. All images are accumulated using a weight of $1/N$. Shadow maps can be used instead of shadow volumes to speed up the algorithm.

Like Monte Carlo ray tracing, instant radiosity was extended to use bidirectional techniques. Bidirectional instant radiosity (Segovia et al. 2006) uses both eye and light subpaths to generate the virtual point lights which, after being resampled based on their contributions, are used in the rendering step. Generating the virtual point lights bidirectionally increases efficiency, as only the VPLs that are significant from the perspective of both the light sources and the camera contribute to the final image. Similarly, Metropolis instant radiosity (Segovia et al. 2007) generates relevant VPLs by mutating paths sampled from the camera to the light sources. The virtual point lights are sampled via perturbations (Veach and Guibas 1997), which yield a faster exploration of the path space and consequently a better convergence rate. Figure 2.21 illustrates the generation of VPLs in bidirectional and Metropolis instant radiosity.

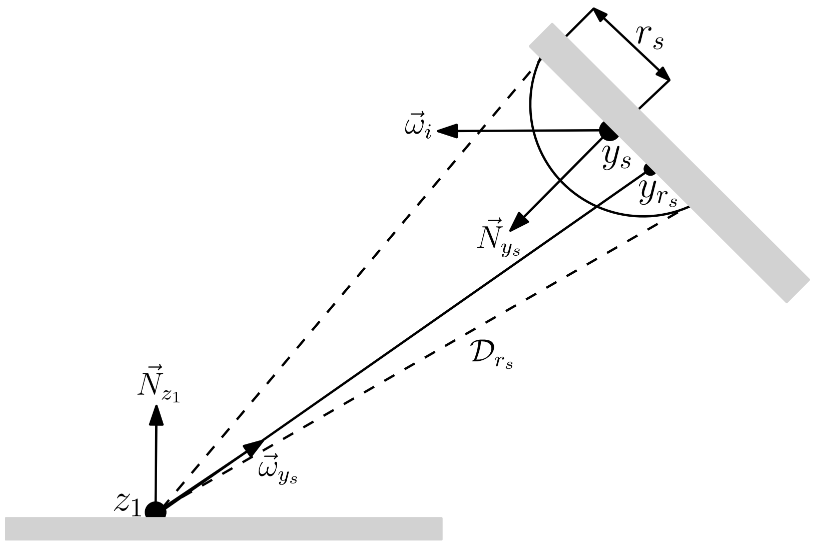


***Figure 2.22**: The geometry that underlies the radiance estimation at primary sample z_1 via a virtual spherical light centred at y_s (amended from Hašan et al. 2009, p. 3).*

However, these methods work under the assumption of diffuse light transport and they suffer from artefacts related to the connecting vertices. As expressed by the radiosity equation, a VPL is connected to an eye subpath through the standard connecting edge:

$$f_r(y_{s-1} \rightarrow y_s \rightarrow z_1)G(y_s \leftrightarrow z_1)f_r(y_s \rightarrow z_1 \rightarrow z_0)$$

Whenever the G factor or the contribution of the BRDF is large compared to the path probability density, artefacts will appear due to the path having a larger contribution than expected. The squared distance $||y_s - z_1||^2$ in the denominator of the geometric factor generates a spike in the rendered image, whenever a VPL lies very close to a primary sample. Due to the small distance, the path contribution will be inflated. A similar effect is produced in the case of a glossy BRDF with a substantial contribution in the direction of the primary sample. However, diffuse BRDFs do not generate spikes.

To eliminate the spikes induced by the geometric factor, the latter can be clamped to a given constant. Yet, the use of a clamping constant introduces bias (equation 2.56).

Virtual spherical lights (Hašan et al. 2009) remove the spike artefacts. Instead of using VPLs to evaluate the radiance contributions, the virtual spherical lights technique considers a disk of radius r_s around each VPL and estimates the radiance of a primary sample by using a cone of incident directions $\mathcal{D}_{r_s}$. Figure 2.22 illustrates these concepts.

By integrating over the cone of directions that travel from the primary sample to the sphere of radius r_S centred at the virtual point light, the proposed method eliminates the geometric spikes. It also renders glossy surfaces, as the BRDF induced spikes are effectively removed through the integration over the cone of incident directions. The integration operation acts as a low-pass filter, smoothing any glossy lobe of the BRDF.

Like the density estimation kernel used in photon mapping, the region around a VPL has two major impacts on the algorithm. Firstly, the radius parameter introduces bias. The algorithm is not consistent, as the radii are always chosen using the local density of virtual point lights. Specifically, the 10 nearest neighbours are searched and the radius r_S is set to a user-defined multiple of the resultant search radius. Secondly, some of the glossy details are blurred, due to the directional integration and area averaging.

Lightcuts (Walter et al. 2005) improve the simulation of indirect illumination effects by clustering point light sources. Each type of light source is discretized using either point, oriented or directional light sources. These light sources are grouped based on their proximity and orientation and a binary light tree is built using a bottom-up approach that progressively combines lights and/or clusters. The clusters are created with minimum sizes and representative light sources are randomly chosen based on the intensities of the children. The image is rendered by adaptively selecting for each point sample a lightcut, which is a set of tree nodes such that every path from the root to the leaf contains a single node from the lightcut. The lightcuts are selected by computing a per cluster error bound and by replacing a node with its children whenever the error bound is larger than the product between a perceptual error (2%) and the total radiance estimate. Radiance is estimated for each node by multiplying the BRDF and geometric factor of the representative light source with the sum of the intensities of all the light sources in the cluster. The cluster error bound is determined by computing upper bounds on the BRDF, geometric and visibility terms of the cluster and multiplying them with the cluster intensity. The amount of error depends on the similarities between the BRDFs, geometric factors and visibility terms associated with the light sources of the cluster. Figure 2.23 displays a light tree with three lightcuts.

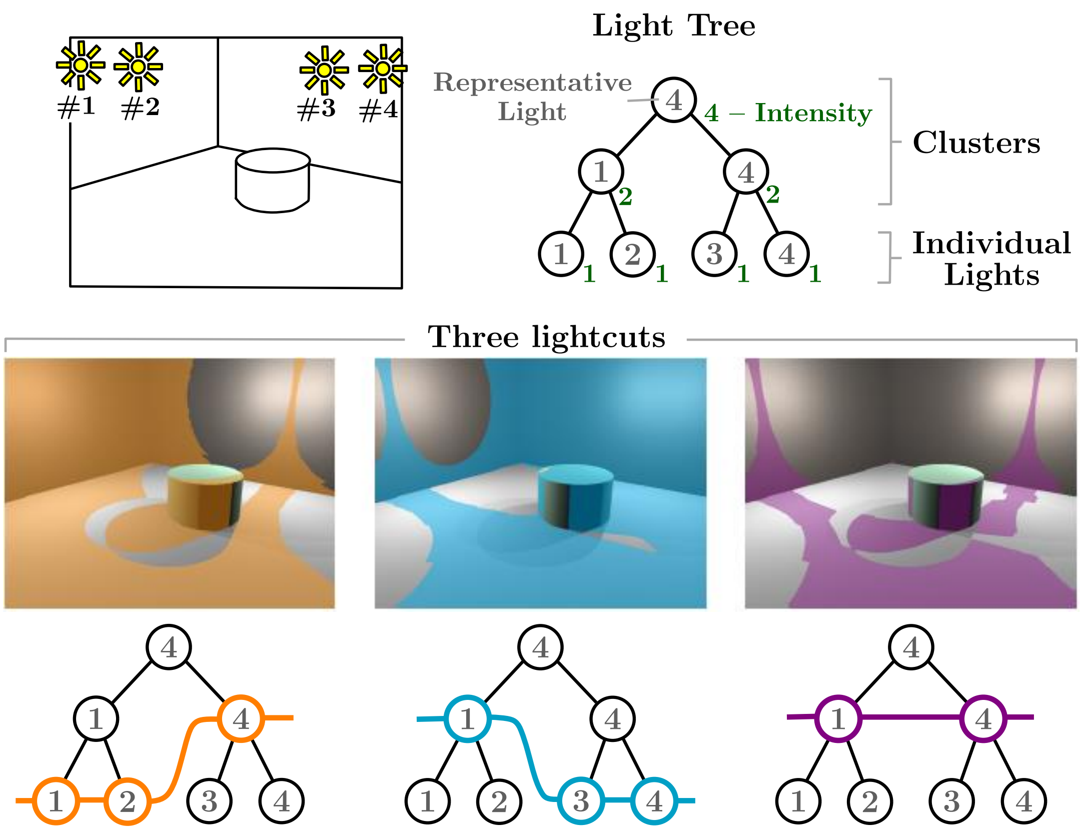
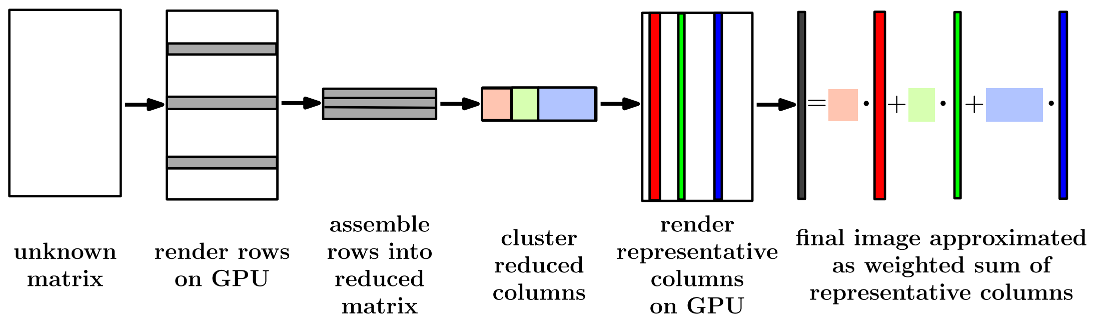


***Figure 2.23**: A binary light tree and three lightcuts. The colours in the images above the lightcuts highlight the regions where the error is small (Walter et al. 2005, p. 3).*

The lightcuts algorithm can evaluate a considerable number of virtual point lights with sublinear cost and thus it can simulate indirect illumination with more details. By employing a less conservative approach, reconstruction cuts exploit the coherence in illumination and further reduce the shading costs and the aliasing problems of the lightcuts. The gist of the reconstruction cuts is to compute lightcuts sparsely over the image and shade the rest of the image by interpolating their illumination information. However, both techniques are subjected to the limitations of the underlying instant radiosity algorithm. Neither lightcuts nor reconstruction cuts can render illumination effects that cannot be simulated with instant radiosity. Moreover, delta components are processed through recursive ray tracing. Lightcuts introduce errors by clustering light sources, but ensure these errors are statistically uncorrelated through the random selection of the representative light sources. In addition, reconstruction cuts may miss features which are not captured by the interpolated lightcuts. Finally, not all types of BRDFs and light sources (e.g. spotlights) are supported by the lightcuts framework.



unknown matrix
render rows on GPU
assemble rows into reduced matrix
cluster reduced columns
render representative columns on GPU
final image approximated as weighted sum of representative columns

***Figure 2.24**: Matrix row-column sampling algorithmic steps (Hašan et al. 2007, p. 3).*

The matrix row-column sampling for many lights (Hašan et al. 2007) reformulates the many-light problem as a large matrix of sample-light interactions, where every column represents all the samples illuminated by a light and every row represents a sample illuminated by all the lights. The sum of the matrix columns gives the ideal image and it is approximated by sampling rows and columns of the matrix on the GPU via shadow maps. The input to the proposed algorithm is the set of samples and the set of virtual point lights, which are determined via ray tracing and instant radiosity respectively. Shadow maps could also be used to generate the samples, while reflective shadow maps (Dachsbacher and Stamminger 2005) could be used to determine one-bounce virtual point lights. The image is divided into blocks and a row is randomly selected from each block via shadow mapping, which is utilized to evaluate the contributions of all lights to a sample by computing the depth map from the sample location. The selected rows are packed into a matrix and used to cluster the reduced columns of the obtained matrix. The columns are reduced since they do not contain all the samples. The clustering selects a given number of reduced columns as cluster centres based on a probability that is proportional to the sum of the distances between the considered column and the rest of the columns. The distance between two columns regards the degree to which two lights dislike being in the same cluster. That is, columns that are farther away from the others are more likely to be selected as cluster centres. If a column is selected multiple times, its weight is increased inversely proportional to its probability. After identifying all the requested cluster centres, every remaining column is assigned to the nearest cluster based on its distance. A representative full column is selected from each cluster with a probability proportional to the reduced column norm. Finally, the weighted representative full columns are accumulated using shadow maps. Figure 2.24 summarizes the main steps of the matrix row-column sampling algorithm.

By using efficient rasterization, the matrix row-column sampling avoids using ray tracing and provides fast user feedback. Unlike lightcuts, which adapt clustering to each sample, it uses global clustering to exploit the GPU capabilities and obtain up to $30\times$ speedups. However, the approach suffers from shadow mapping artefacts (e.g. shadow bias), clamping artefacts (e.g. corner darkening) and slight temporal artefacts.

Virtual point light techniques benefit from fast executions. However, they do not simulate a comprehensive range of illumination effects. For example, generic BSDFs and $S^{+}DS^{+}$ paths are not supported despite recent advancements (Hedman et al. 2016).

2.5.8 The choice of algorithmic foundation

The practical goal of the current work is to implement a light transport framework that simulates a wide range of illumination effects in conditions of dynamic geometry. The review on the global illumination classes identified several algorithms that satisfied the prerequisites established for the development of the light transport framework.

Monte Carlo ray tracing, photon mapping and unified estimation were all identified as classes of algorithms that efficiently simulate the needed range of illumination effects. Bidirectional path tracing prevailed as the best basis for the light transport framework.

Compared to path tracing, bidirectional path tracing is a generalization benefiting from better exploration of the path space and better convergence for certain configurations.

Compared to Metropolis light transport, bidirectional path tracing benefits from simplicity and lack of bias. Unlike the former algorithm, it explores the path space globally and does not require good mutation strategies to ensure that the random walk does not get caught in a subregion of the path space. Furthermore, it does not suffer from start-up bias, image plane sample stratification or path correlation artefacts. Consequently, bidirectional path tracing circumvents the difficulties associated with finding good ergodic mutations and generating a sufficiently large collection of paths.

Compared to progressive photon mapping, bidirectional path tracing has the advantage of being unbiased and not merely consistent. The lack of bias enables the computation of the error based on variance alone (equation 2.58). This feature causes the images generated with bidirectional path tracing to be used as references in the result analysis of other algorithms. Moreover, bidirectional path tracing does not suffer from blur caused by the density estimation, nor from other geometric biases (Herzog et al. 2007).

Being formulated in the path integral framework, bidirectional path tracing easily supports the use of multiple importance sampling. Combining paths, sampled from various probability densities, is the key to the robustness of bidirectional path tracing. The unified estimation algorithms reformulate photon mapping in the path integral framework, so that they can take advantage of the powerful variance reduction technique and improve the efficiency of the light transport simulation with an enhanced set of sampling techniques. These algorithms are more efficient at simulating effects like S^+DS^+ light transport, but they are still only as efficient as the algorithms that they combine. Phenomena for which the underlying algorithms do not provide good sampling strategies, are not simulated efficiently. Also, unified estimation algorithms are only consistent. For these reasons, bidirectional path tracing prevailed as the best development foundation. It might be less efficient at simulating certain effects, but its advancements could be combined in the style of unified estimation. Thus, the current work focuses on improving bidirectional path tracing. This algorithm represented the starting point for the development of the path manipulation strategies. The rationale of the path manipulation strategies is to inject dynamism in Monte Carlo simulations, by advancing the algorithm itself, without necessarily relying on hardware techniques. The core idea is to extend the use of paths to the temporal domain, through an apparatus of sampling and reuse strategies. By reconstructing and reusing subpaths, the resultant path manipulation algorithm avoids regenerating the entire path collection, reduces the computational load of the original algorithm and supports scene dynamism. The next section compares the current work with the recent advances in light transport simulation.

2.6 Path manipulation strategies and related work

Path sampling techniques constitute the foundation of Monte Carlo ray tracing. Using the information localized at path vertices, sampling techniques construct paths with different probability densities. On the one hand, the punctiform nature of the sampling techniques decouples a Monte Carlo algorithm from the specificity of the input. On the other hand, the sampling techniques determine the efficiency with which the measurement contribution function (equation 2.44) is estimated. In other words, each individual technique efficiently samples a specific group of illumination effects, by taking into account different factors of the measurement contribution function.

The set of sampling techniques that a Monte Carlo ray tracing algorithm uses, defines the efficiency of the light transport simulation. The simplest form of Monte Carlo ray tracing, explores the path space by generating unidirectional paths. Representative of this modus operandi are the shooting algorithms, which trace light particles up to the camera (Dutré and Willems 1995). In a reciprocal manner, gathering algorithms (e.g. naïve path tracing) scatter importance in the scene until a light source is intersected. The efficiency of unidirectional approaches can be moderate, because few paths tend to intersect the camera or the light sources, causing the solution to converge slower.

Unidirectional approaches improve their efficiency by utilizing additional sampling techniques. For instance, next event estimation is used by both shooting and gathering algorithms. This estimator assesses the measurement contribution function at each path vertex via a direct connection to either the camera or a light source. In this way, more sampling techniques explore the path space and more illumination effects are sampled.

A more efficient tier of Monte Carlo ray tracing algorithms exploits better the path sampling techniques. Metropolis light transport (Veach and Guibas 1997) uses path mutations to explore regions with high contribution and thus expedite the simulation of difficult illumination scenarios. Bidirectional path tracing generates and combines paths based on a family of sampling techniques. By generating paths dually from the light sources and the camera, bidirectional path tracing exploits the universality of the path generation process. Moreover, the contribution of each path is evaluated based on all the sampling techniques that could have generated that path. That is, multiple importance sampling optimally combines the paths sampled with different techniques.

By using a set of sampling and reuse strategies, the current work ports bidirectional path tracing to the temporal domain and facilitates scene dynamism. Conventionally, sampling techniques are used to generate and evaluate the contributions of paths. Thus, their potential is limited to the path generation-evaluation cycle. Discarding paths after the evaluation of their contributions, implies a static usage of paths and of sampling techniques. Moreover, the temporal dimension is supressed with this approach. The path manipulation algorithm uses sampling techniques in a temporal aware approach that supplants the static manipulation of paths with a generation-evaluation-reuse cycle.

2.6.1 Improved efficiency via sampling-oriented techniques

Despite the gradation of efficiency via sampling techniques, Monte Carlo ray tracing suffers from being computationally expensive. On the one hand, the execution costs can be credited to the numerous sampling operations, visibility tests and path contribution evaluations. On the other hand, the stochastic nature of the Monte Carlo integration causes variance (Veach 1998, p. 39). The convergence rate of an unbiased estimator is $O(1/\sqrt{N})$. The immediate implication is that acceptable levels of variance require a significant number of paths to be generated, whence the high execution costs.

To improve efficiency, Monte Carlo algorithms use various cost reduction techniques.

One approach refers to reducing the number of traced paths, by reusing them across different measurements. For example, Metropolis light transport performs mutations by reusing as much path information as possible. Bidirectional path tracing can reuse light subpaths across connections with different eye subpaths. Also, additional samples can be efficiently generated from the same complete light transport path, by joining each prefix of the light subpath to each suffix of the eye subpath (Veach 1998, p. 300).

Another means to achieve better performance is to reduce the cost per individual path. Two extensively used, complementary techniques that serve this purpose are Russian roulette and splitting (Arvo and Kirk 1990). Russian roulette reduces the number of scattering events, at the cost of introducing extra variance. Conversely, splitting can reduce variance and increase performance by adapting the number of scattering events (i.e. the density of samples) to the estimated contribution of the light transport path.

Variance reduction techniques represent another approach to reducing execution costs. The most pervasive method is multiple importance sampling, which reduces variance by optimally combining samples from multiple techniques. Veach (1998) provides a comprehensive discussion on all four major variance reduction categories, namely analytic integration, uniform sampling, adaptive sampling and correlated estimators.

Algorithms such as vertex connection and merging (Georgiev et al. 2012) and unified path sampling (Hachisuka et al. 2012) improve the efficiency of the light transport simulation by enhancing the regular set of bidirectional sampling techniques with sampling techniques derived from the photon density estimation. The augmented set of sampling techniques explores better those types of light transport paths that involve specular surfaces, improving thus the low convergence rates of the individual methods.

Probabilistic connections for bidirectional path tracing (Popov et al. 2015) improve the efficiency of the original algorithm by establishing bidirectional connections based on the contribution of a discrete set of light subpaths. At each iteration one eye subpath is generated per pixel and M light subpaths are stored. For a small subset of eye subpaths, sampled uniformly on the image plane, the proposed method computes and stores probability mass functions. The probability mass functions are computed by normalizing the contribution of each light subpath with the sum of the contributions of all of the M light subpaths. For eye subpaths that do not store records of probability mass functions, the approach finds the closest located records, interpolates them and then uses the CDF inversion method to sample connections and evaluate contributions.

Using Gaussian mixture models, Vorba et al. (2014) address the difficulty manifested by Monte Carlo ray tracing in rendering scenes with complex visibility. In a training pass, the proposed method learns the distributions of the light and importance particles and uses them, in the rendering pass, to augment the sampling of emission and scattering directions with an estimate of the equilibrium radiance or importance. That is, the equilibrium importance is used to guide light subpaths towards the camera, whereas the equilibrium radiance is used to guide eye subpaths towards light sources. Guiding subpaths towards each other increases the probability of connecting them with non-zero contributions. A higher number of valid contributions reduces variance.

Gradient-domain bidirectional path tracing (Manzi et al. 2015) uses finite difference gradients to reduce variance and consequently rendering time. Following the outline of gradient-domain path tracing (Kettunen et al. 2015), the bidirectional algorithm generates a number of base, full transport paths for each pixel. The base eye or light subpaths are then shifted, using manifold exploration (Jakob and Marschner 2012), to the horizontal and vertical neighbouring pixels. The contribution difference between a shifted path and its base homologue is stored in a gradient image. In a post-processing pass, the regular and the gradient images are combined via a screened Poisson solver.

Gradient-domain path reusing (Bauszat et al. 2017) improves the convergence rate of gradient-domain path tracing by reusing eye subpaths in the computation of the image gradients. After splitting the image into tiles and generating a single eye subpath per pixel, the proposed algorithm shifts each base subpath to all the pixels in the tile via a generic path reusing function. The reusing function connects the primary sample of a pixel with the higher-order sample of a subpath generated for another pixel, by shifting the subpath such that a connection with a non-zero contribution can be established between the two samples. The contributions of the shifted subpaths serve to estimate the gradients. The process is repeated for all the requested samples per pixel. The final image is reconstructed from pixel values and gradients via a screened Poisson solver.

Metropolised bidirectional estimators (Šik et al. 2016) take one step further and extend vertex connection and merging with Markov chain Monte Carlo sampling, so as to efficiently process both complex specular transport and complex visibility. Using a standard Monte Carlo sampler, the proposed algorithm generates all the eye subpaths, which are immediately evaluated using next event estimation. The subpaths which intersect a light source are also evaluated using naïve path tracing. The light subpaths are generated with a Markov chain Monte Carlo sampler and evaluated using the remaining vertex connection and merging techniques. In this way, the strength of the bidirectional estimators in handling glossy/specular transport is combined with the versatility of the Markov chain Monte Carlo sampling in addressing complex visibility.

As discussed in chapter 4, the current work implements the bidirectional path tracing algorithm and the path manipulation strategies with path reuse, Russian roulette and multiple importance sampling. However, the gist of the path manipulation algorithm is the extension of bidirectional path tracing towards reusing paths in the temporal domain. As opposed to the previous advancements, the current work does not focus on deriving strategies that increase the efficiency of the sampling process itself. That is, the path manipulation algorithm does not provide additional sampling strategies that improve the simulation of certain difficult light transport paths. It does not reduce variance by guiding subpaths or by establishing probabilistic connections. Neither does it enhance the basic algorithm with tools like image gradients or metropolised sampling.

Instead, it focuses on developing sampling strategies, inherent to the path integral framework, that are capable of exploiting temporal coherence during the rendering process itself. Though improving the performance of the original algorithm, the above methods explore the spatial dimension while ignoring the temporal one. They may store and reuse paths within one iteration of the algorithm, yet new paths are generated between iterations. For example, the probabilistic connections approach stores light subpaths and reuses them across connections with different eye subpaths. Unified path sampling and metropolised bidirectional estimators also reuse paths. However, the path generation-evaluation cycle is repeated between iterations. The current work aims to supplant this static manipulation of paths with a generation-evaluation-reuse cycle.

2.6.2 Accelerated Monte Carlo ray tracing

The capacity stream processing architectures exhibit at delivering high throughput, represents another driving force aimed at reducing the execution costs of Monte Carlo ray tracing. In other words, progressive Monte Carlo ray tracing invested a lot of effort in adapting the Monte Carlo algorithms to the rigours of stream processing technology.

One direction of research involves the development of high-performance ray shooting engines (Parker et al. 2010; Wald et al. 2014). The aim of the ray shooting investigation is to optimally select and efficiently construct and traverse acceleration data structures.

An alternative to ray shooting based on acceleration data structures is divide-and-conquer ray tracing (Mora 2011; Áfra 2012; Nabata et al. 2013; Ravichandran and Narayanan 2013). The latter category of algorithms circumvents the construction of acceleration data structures by correlating the partitioning of primitives with the tracing of rays. The major advantage in doing so is that the partitioning time can be significantly less than the building time of an acceleration structure, saving both precious time and memory especially for applications involving dynamic scenes.

The initial divide-and-conquer algorithm (Mora 2011) employs spatial partitioning to recursively divide the bounding box of a list of triangles along the longest dimension. For scenes with less than 10,000 triangles a median cut schema is used, whereas for the rest of the scenes the split locations are determined by uniformly selecting a 50th of the entire triangle stream and averaging the middles of the triangle spans. A triangle span is the difference between the triangle's maximum and minimum values along the split axis, clamped by the bounding box limits. The intersections of the triangles/rays with the currently evaluated bounding box are used to sort an auxiliary list of indices. All the triangles/rays which intersected the current bounding box are identified via a contiguous range of indices. The index sorting, performed in a breadth-first quicksort manner, avoids moving the larger triangle/ray structures and is designated as filtering.

The tracing of primary rays is optimized by traversing the spatial subdivisions in a front-to-back order and by suppressing the computation of the intersections beyond the first successful intersection. The tracing of rays emanating from common locations (e.g. camera, point or virtual point light sources) is further optimized by tracing conic ray packets instead of individual rays. These optimizations do not apply to incoherent rays, for which the closest scene intersection must be computed using a Z-buffer like approach. When the number of triangles or rays falls below a given threshold, the active rays are traced against the subdivision triangles via naïve ray tracing. Otherwise subdivision, intersection, filtering and traversal continue to be executed recursively.

Ravichandran and Narayanan (2013) parallelize the divide-and-conquer ray tracing algorithm to benefit from the streaming capabilities of the GPU. The parallel approach exhibits increased performance for primary rays, incoherent rays and scenes with viewpoints located outside the bounding box. Its performance decreases relative to that of the CPU-based algorithm for viewpoints located inside the bounding box, for which the latter algorithm benefits from early ray termination and conic packet tracing.

Áfra (2012) succeeds in optimizing the CPU-based algorithm for incoherent rays by efficiently filtering rays, using higher quality object partitioning and exploiting wider SIMD sets. By sorting the rays instead of their indices, Áfra (2012) achieves linear memory accesses and avoids the cache misses incurred by the initial algorithm. This filtering suits incoherent rays, increasing the ray traversal performance by at least 25%. By adopting object partitioning over spatial partitioning, the list of primitives is divided into disjoint lists. The median cut and surface area heuristic split the list of primitives along the longest dimension of an axis aligned bounding box. The surface area heuristic is selected whenever the ratio between the active rays and the current triangles exceeds a given threshold. Primitives are filtered by sorting the associated list of indices. The traversal order is also adapted to incoherent rays by considering the axis on which the two child bounding boxes are farthest apart, the direction sign of the first active ray along that axis and the closest child along the identified direction. Finally, the advanced vector extensions are faster than the streaming SIMD extensions.

Nabata et al. (2013) further accelerate the tracing of rays by exploiting the distribution of rays and a cost metric which avoids inefficient ray filtering. Like Áfra (2012), the proposed algorithm uses a bounding volume hierarchy represented as a binary tree. The scene bounding volume is subdivided into several bins, each with its left and right volumes bounding two disjoint sets of triangles. When the rays are over 1000, some of them are randomly selected and traced against every pair of left-right volumes to determine the cost function, the ray distribution in the bounding volume and the traversal order of the child volumes. The cost function is assessed using the surface area heuristic, with the intersection probabilities for the left and right volumes substituted with the intersection ratio between the number of rays intersecting that precise volume and the total number of randomly selected rays. The smallest cost function identifies the bin to be investigated. The traversal order is determined by selecting the child volume with most of the selected rays closer to it. The closest rays are determined by intersecting the selected rays with each child volume. Ray filtering is made efficient by suppressing it whenever the intersection ratio exceeds a certain threshold. The proposed algorithm exhibits superior performance for all types of ray when compared to Áfra (2012) and for incoherent rays when compared to Mora (2011).

A challenge in simulating light transport on stream processing architectures is precisely the incoherence of scattered rays. One technique for accelerating ray tracing is to take advantage of the coherence of certain rays and group them into packets. Rays originating at the same location (e.g. primary rays, rays emanating from point light sources or soft shadow rays cast from a sample) exhibit sufficient coherence to be traced in packets and benefit from efficient SIMD instructions and packet-based culling. However, most light transport rays are incoherent as they originate from different scene locations, tend to go in all directions and cover large scene portions. Boulos et al. (2007) demonstrate that higher-order rays can be assembled in general packets based on a common property. Runs trace all the higher-order rays that share a property (e.g. object identifier, material, geometric proximity, surface orientation) and correspond to adjacent primary rays. Groups trace all rays with a common property, while ray types assemble shadow, refraction and reflection rays in 3 distinct packets. Ray types prevailed over the other packing strategies, due to low overhead and less sensitivity to ray ordering than runs. Groups were considered too general to be viable.

Boulos et al. (2007) also identify the bounding volume hierarchy as robust to false negatives in ray-box intersections, deformable primitives and spread in ray packets. Performance is increased beyond the SIMD gain by combining general packets with a bounding volume hierarchy traversal that benefits from early ray termination and ray packet culling. Run packets were reported to perform an increased number of ray-box intersections at higher bounce depths, reducing performance by 10-20% as compared to ray type packets. Thus, memory accesses and higher-order rays dictate performance.

Aila and Karras (2010) discuss architectural solutions that reduce memory traffic for efficient tracing of incoherent rays. The proposed approach divides the bounding volume hierarchy into treelets and queues rays with each treelet to reduce cache pressure. A custom architecture with 16 processors fetches rays from an input queue in the DRAM and fills warps of 32 threads. During execution each ray fetches two child volumes, traverses the closest intersected one and stores the other one's index on a per-ray traversal stack. When the non-terminated rays in a warp fall below 50%, the rays are compacted by storing the terminated ones in memory and copying the data of the non-terminated ones to the warp that is currently being filled. Compaction increases the non-terminated rays from 25% to 60-75%, suggesting over 2× speedups. Small differences in the memory traffic for rays sorted randomly (worst case) and relative to an origin-direction space-filling curve (best case), led to the conclusion that ray order is irrelevant. The explanation is that, for increasingly complex scenes, rays tend to visit nodes further apart in the tree. By storing the per-ray traversal stack in memory and accessing it via a stack top cache, the traversal stack traffic is eliminated and the total memory traffic is approximately halved. Treelets are built from the leaves up by optimizing the total surface area of their roots and rays are stored in the queue of the treelet whose boundary they encounter. Two types of schedulers decide when a treelet queue has accumulated sufficient rays to be traced. A lazy scheduler assigns a processor with a currently empty queue to the queue with the highest number of rays. A balanced scheduler assigns processors based on the number of processors a queue requests and on the number of rays. The requested processors grow linearly from the moment a treelet queue exceeds a target size and until it reaches the double target size.

Aila and Karras (2010) prove that a lazy scheduler works best with treelets that reflect the L2-cache size, while a balanced one works best with treelets that fit the L1-cache size. Unlike the lazy scheduler, the balanced one exhibits increased memory traffic, but makes practically feasible assumptions about the speed of the L2 cache. A key advantage of ray packing is the ability to accommodate the requested nodes/triangles in the L1-cache. The treelet queues are implemented in external memory, with dynamic resizing to circumvent deadlocks and pre-emption operations. A queue is bypassed and memory traffic is further reduced by dispatching a ray to another processor that is already bound to that specific queue. A ray can also be forwarded to a processor which in the previous two executions was bound to the queue that should have stored the ray. The rationale is that rays from a previous queue can still be in execution even after binding the processor to another queue and treelets about to be completely traversed have fewer rays, allowing the processor to trace rays from multiple treelets. Bypassing reduces memory traffic by 10% if only the current queue is considered and by 20% if the previous two queues are allowed. Aila and Karras (2010) outline interesting future refinements, but identify auxiliary traffic (rays, queues, stacks) as the main challenge.

Thus, another area of interest within progressive Monte Carlo ray tracing is concerned with fully utilizing the available resources. The full use of the streaming capabilities is synonymous with minimizing the divergence of the execution flow, which is generally achieved by designing operations to access contiguous blocks of memory and by imposing a maximum depth on the traced paths (van Antwerpen 2011a). Having all paths traced up to a certain length avoids the suboptimal sharing of processing units between working and idle threads. Yet, in proceeding as such the efficiency of the Monte Carlo algorithm itself is affected, since paths are traced up to the imposed length regardless of the contribution they bring with each scattering step. Novák et al. (2010) try to remedy this limitation by means of a two-stage algorithm, which in its first phase still traces paths up to a user predefined length, but which in its closing phase continues tracing the terminated paths up to a length defined in terms of the average scene reflectance. Being essentially a path generation strategy, the proposed technique lends itself to both path tracing and bidirectional path tracing.

In addition to compacting operations with the purpose of accessing contiguous blocks of memory and benefiting from full memory bandwidth, progressive Monte Carlo algorithms also face, on some processing units such as the GPU, restrictive memory capacities. Memory limitations constitute a major challenge in implementing some of the Monte Carlo algorithms, like bidirectional path tracing, on the GPU. State-of-the-art solutions such as streaming bidirectional path tracing (van Antwerpen 2011b) and light vertex cache bidirectional path tracing (Davidovič et al. 2014) address this problem using pure GPU implementations. Combinatorial bidirectional path tracing (Pajot et al. 2011), avoids some of the GPU limitations by generating all the eye and light subpaths on the CPU and then connecting all their vertices on the GPU. Exhaustively combining subpath vertices reduces the sampling costs via path reuse, limits the memory footprint to an adequate size and ensures a suitable GPU occupancy.

The advent of stream processing technology and the development of progressive Monte Carlo algorithms, considerably improved the balance between accuracy and performance. However, research in this area is dictated by the necessities that arise with porting the offline algorithms on the designated architectures. That is, the focus is on developing structures and mechanisms that adapt the original Monte Carlo algorithms to the design, capabilities and limitations of the hosting platform. For example, the discussed progressive algorithms harness the power of the GPU to attain fast executions. Yet, none of them focus on scene dynamism and temporal coherence.

The methods that do exploit temporal coherence refer mainly to accelerating classic ray tracing. Research in this area either produced parallel ray tracers that fully engage the resources of multi-processor systems or created techniques that optimally process acceleration structures and exploit ray coherence. For instance, Martin et al. (2002) use a 32-processor system to reproject anterior primary samples onto the image plane and guide the ray tracing of the current frame in a way that reduces flickering and popping.

Wald et al. (2003) ray trace dynamic scenes on a cluster of dual-core machines. For static objects and for groups of primitives subjected to the same transformations, i.e. to hierarchical motion, their method builds once the local binary space partitioning trees and then computes the intersections of these objects with the rays transformed to the corresponding local coordinate systems. Objects with unstructured motion, i.e. that move independently of all other objects, are stored separately and their acceleration structures are rebuilt for each frame. Similarly, Chapman et al. (1991) transform a ray to object space and compute its intersections with the geometry for the entire animated sequence. Lauterbach et al. (2006) make no assumptions about the motion of objects and dynamically update or rebuild the scene bounding volume hierarchy. The proposed method uses a heuristic, based on the surface area ratio between a parent node and its children, to determine the degradation level beyond which the hierarchy can no longer be updated and must be rebuilt. The method also exploits ray coherence by generating ray packets and traversing the nodes of the hierarchy if at least one ray intersects them.

Wald et al. (2007) report on a variety of techniques, which explore the principal research directions assumed in ray tracing dynamic scenes. The survey analyses the design decisions and trade-offs involved in selecting the best suited acceleration structure for a given scene, motion type and ray characteristics (e.g. number and type). However, the focus is on mechanisms that quickly build or update the acceleration structures and on algorithms that efficiently exploit ray coherence at traversing them.

Recent advances (Guntury and Narayanan 2010; Guntury and Narayanan 2012; Günther 2014; Nah et al. 2015; Pérard-Gayot et al. 2017) do not seem to depart from the mainstream research, as they continue to explore the avenues of performant builds of acceleration structures and coherent ray traversal. As noted by Wald et al. (2002), Monte Carlo ray tracing entails a different level of complexity than classic ray tracing does. By computing the radiance equilibrium of a scene, Monte Carlo ray tracing poses issues well beyond the coherence of primary and secondary rays. Load balancing, contained synchronization, higher-order ray incoherence, decomposition of the global illumination computations into independent tasks and their efficient executions, are some of the aspects that a fast ray tracing global illumination system must consider.

In fact, Wald et al. (2002) combine distributed ray tracing, instant radiosity and photon mapping techniques in a parallel algorithm that relies on restrictions to compute the global illumination solution. The illumination information, subjected to constraints like maximum path length and specific surface properties, is recomputed every frame, with the same pseudorandom numbers and low-discrepancy sequences, to avoid flickering.

Techniques based on rasterization can evaluate the indirect illumination arising from a limited number of bounces in real-time. For instance, bidirectional path tracing via rasterization (Tokuyoshi and Ogaki 2012) generates one-bounce light subpaths using reflective shadow maps, determines the primary rays via the geometry buffer, extends them with one edge using global ray-bundles and establishes their connections with the light subpaths via shadow maps. However, such techniques introduce approximations beyond the limitation of the subpath lengths. Bidirectional path tracing via rasterization exclusively assumes point light sources and diffuse or moderately glossy BRDFs. The global ray-bundles do not sample according to the BRDF causing spike artefacts on glossy surfaces, which are smoothed by clamping the roughness parameters. However, important cues may be lost by blurring the high-frequency illumination details. The limited resolution of the ray-bundles may also cause artefacts. The geometric factor in the probability density of a two-bounce path generates spikes too, which are solved only biasedly via clamping. Multiple bounces can be generated for small scenes at the expense of memory. By limiting the scene configurations and by approximating the illumination computations, such rasterization techniques distance themselves notably from the current work and will not be further considered. Ritschel et al. (2012, p. 14) provide an ampler discussion on the rasterization-based global illumination techniques.

The current work takes a different route to facilitating scene dynamism in Monte Carlo ray tracing. Instead of conforming a Monte Carlo algorithm to the rigours of a specific architecture or efficiently refitting and traversing acceleration data structures, the path manipulation algorithm exploits the temporal coherence of entire light transport paths.

2.6.3 Temporal coherence and path reuse

An extensive volume of work (Damez et al. 2003; Tawara et al. 2004) uses the temporal coherence in lighting distribution as a premise for circumventing redundant global illumination computations in dynamic environments. The strategies that exploit this temporal coherence vary across the different classes of global illumination algorithms. Yet, the common goal is to preserve as much of the already computed lighting solution as possible. For example, temporal gradient-domain path tracing (Manzi et al. 2016) reuses the same random variables, that generated the base paths in the previous frame, to sample the base paths in the next frame. The shifted paths are computed from the base homologues and the spatial (base-shifted), temporal (base-base) and mixed (shifted-shifted) gradients are determined from the base and temporal offset frames.

Knecht (2009) reduces the cost of evaluating illumination from virtual point lights by diminishing the number of VPLs through the reuse of illumination information from the previous frame. Being a real-time version of instant radiosity, the proposed algorithm uses imperfect shadow maps (Ritschel et al. 2008) to sample multi-bounce VPLs with a reduced fill rate. The first tier of VPLs is sampled using cube maps for point light sources and importance maps for spotlight sources. In the latter case, point samples are first generated from a uniform distribution and then hierarchically warped (Clarberg et al. 2005) according to the levels of an importance map. The importance map is created with the geometry buffer of the spotlight using the light intensity, light colour, surface colour and specular power. The final positions of the point samples are used to lookup the surface information, in the light geometry buffer, that will generate the VPLs. The other VPLs are sampled by discretizing the scene into a point cloud and assigning the point samples with the largest contributions to existing VPLs. The assignment operation is implemented as an imperfect shadow map test, which selects for each VPL the visible point sample with the largest specular contribution. Yet, the assignment of one point sample to each VPL generates a suboptimal distribution and the disposal of the less dominant point samples introduces bias. Furthermore, only primary samples are generated from the camera. These primary samples are shaded by dividing the camera geometry buffer into several tiles and allotting each VPL to a tile.

Each tile displays the entire scene using a subset of the camera geometry buffer pixels. The contribution of a VPL to its tile is evaluated using its imperfect shadow map and is additively written to the accumulation buffer. Once shaded, the tiles are merged by interpolating the current illumination with the illumination from the last frame. The recycling degree of the prior illumination information is determined by a per-pixel confidence value dependent on the sample position change, normal change and difference between the current and previous illumination values. The illumination discrepancies, owed to discontinuities in the geometry buffer, are smoothed with a geometry-aware box filter. The direct illumination, computed in a 2D post-processing step, is combined with the indirect illumination and is tone mapped to yield the final image. By lighting only a pixel subset with each VPL and by smoothing illumination over time, shading costs and temporal aliasing are reduced. Yet, temporal smoothing artefacts occur when objects move fast or the light source is suddenly occluded. The typical instant radiosity spikes are suppressed by biasedly clamping the VPL-sample distances to a minimum value of 1. Other problematic issues of this approach regard limited number of bounces, lack of heuristics for optimal parameter selection, loss of indirect shadows due to few point samples, temporal artefacts when less than 256 VPLs are used, texture lookup limitation for deformable geometries and potentially detrimental impact of the lookup limitation on temporal coherence. The current work takes a fundamentally different approach to exploiting temporal coherence. It extends bidirectional path tracing to the temporal domain by reconstructing and reusing random-length light and eye subpaths via platform-independent sampling techniques. Hence, the problematics inherent to GPU-based instant radiosity methods are avoided.

Lai (2010) provides a broad discussion on the temporal-aware algorithms developed in various global illumination research areas. Recent techniques, such as the sequential Monte Carlo instant radiosity (Hedman et al. 2016), confirm the renewed interest in the temporal processing of global illumination. However, there is little development in the area of path reuse. Few techniques approach temporal coherence by extending a Monte Carlo method without recourse to additional tools like image-space gradients. By directly processing light transport paths, path reuse techniques constitute the immediate context of the current work and are examined for the rest of this subsection.

Path reuse techniques accommodate scene dynamism by recycling path information. Across different solutions, the degree of information preservation ranges from primary samples (Havran et al. 2003) to parts of light subpaths (Sbert et al. 2004a), parts of eye subpaths (Bekaert et al. 2002) or subpath parts of both types (Sbert and Castro 2004). Nevertheless, the existent path reuse approaches work on rather limiting assumptions.

Havran et al. (2003) present a spatio-temporal architecture for animation rendering, which assumes the animation paths, for the camera and moving objects, are known a priori. On the one hand, the presupposition of animation paths precludes the use of the proposed framework with applications that require direct interaction with the scene, as is the case of the simulation and design software used in commercial industries (e.g. optical and photometric simulation within CAD/CAM tools and light management via virtual reality systems). On the other hand, the approach operates primarily on paths with diffuse primary samples and applies a weighting heuristic that biases the solution, i.e. the contribution weights do not reflect the probabilities used to sample the paths.

Méndez-Feliu et al. (2006) remedy both problematics by normalizing the probabilities of the original samples relative to the pixels onto which these samples are reprojected. The weighting heuristic is also computed based on the probabilities with which a given sample is reprojected on all other possible pixels. Though constructing a general and unbiased Monte Carlo estimator, the proposed algorithm handles only camera motion.

Similarly, Bekaert et al. (2002) accelerate path tracing by constructing an unbiased estimator that reuses eye paths. The reused paths are formed by reconnecting primary samples to secondary samples that appertain to paths generated through nearby pixels.

Conversely, Sbert et al. (2004a; 2004b) restrict the scene dynamism to moving light sources. Sbert et al. (2004a) reuse light subpaths by reconnecting the moving light to the primary samples of the light subpaths generated in the previous and subsequent frames. Sbert et al. (2004b) assume punctiform light sources moving in a static scene and compute an unbiased estimator like the one proposed by the former approach.

Both techniques use a priori knowledge, about the light source animation paths, to generate all the frames (pre-processing phase) and then reuse paths across frames from both preceding and subsequent frames (animation phase). To avoid the restriction of a priori known animation paths and thus accommodate interactive applications, the latter approach restricts the reuse of paths to preceding frames. Nevertheless, both algorithms remain limited from a scene dynamism standpoint. By fixing objects and camera, neither method offers a generic solution for reconstructing and reusing paths.

Sbert and Castro (2004) reuse both light and gathering subpaths. Light subpaths are reused as discussed by Sbert et al. (2004a). However, gathering subpaths are limited to a single edge and their reuse implies accruing, at every sample, the radiosity of the preceding and subsequent frames. Furthermore, the restriction of dynamism to light source movements and the assumption of a priori known animation paths are retained.

Méndez-Feliu and Sbert (2006) combine light and camera animations to produce an array of frames that can be collated in a movie, based on the transformation of either the camera, the light source or both. In the case of camera animation, the proposed method uses the system designed by Havran et al. (2003) to recycle the direct diffuse illumination and the obscurances (Iones et al. 2003) computed for indirect illumination. Specular effects are handled by generating paths normally. For light animation, the direct illumination, specular effects and ambient light are computed for each frame, whereas obscurances are reused. Combined light and camera animations are computed by generating for every camera position the frames associated with all the light positions. The result is a high number of images that reflect the different combinations of light and camera transformations. Consequently, the exploration of the animation space is limited to the set of generated images. The proposed method is also limited by the assumptions of static objects and a priori knowledge about the animation paths.

Lai (2010) extends population Monte Carlo equal deposition energy redistribution with a temporal perturbation to render an entire animated sequence. In a pre-processing phase, the proposed algorithm uses path tracing to generate caustic paths and paths distributed uniformly across the image plane. Some of these paths are used as seeds, while others are used as substitutes in the resampling phase. A given number of spatial and temporal perturbations are applied to every seed path. Each perturbation is either accepted or rejected and the energy of the perturbed/original path is deposited at the associated pixel. The resampling phase excludes well-distributed and low-contribution paths, replacing them with substitute paths so as to maintain a constant path number. At the end of the redistribution-resampling process the residual energy of all processed paths is equally deposited onto a range of frames. The temporal perturbation generates a new path by applying rigid transformations to the first two vertices of the path and then extending the path either through specular bounces or by rigidly transforming the diffuse vertices. The locations of the specular vertices determine whether the temporal perturbation starts at the light source or at the camera. To be feasible, the proposed algorithm is implemented using a client-server paradigm. The server executes the pre-processing phase, creates client tasks, collects client data, updates intermediate results and generates the necessary information for the next iteration. Each client renders a subset of the initial paths via an unbiased strategy that locally adapts the rendering parameters. The initial paths are distributed using their timestamps and rays are traced against a bounding box in the kd-tree only if their timestamps belong to the timestamp interval of that bounding box. To reduce memory pressure, the contributions of the paths that fall within a 20-frame radius from a central frame are stored in memory, whereas the other contributions are stored in buckets on the disk. Besides assuming predefined animation paths, the proposed algorithm also assumes low variation in the lighting condition to ensure convergence in similar iterations. Changes in illumination are processed exclusively by cutting the animation into smaller animated sequences.

The path manipulation strategies reconstruct and reuse paths independently of the subpath or scene dynamism type. Light and eye subpaths are reconstructed generically and regardless of whether the transformations affected the camera, a light source or other scene objects. Moreover, predefined animation paths are not required. Tables 2.1 and 2.2 compare the path manipulation algorithm with the perused temporal methods.

Method	Dynamism	Interactivity	Reuse	Transport	Memory footprint	Artefacts
Primary sample reconnection Bekaert et al. 2002	—	—	*EP intra frame*	$L(S\|D)^{*}E$ $MPL=\infty$	est. $O(EP)$	$\begin{cases}Unpleasant\ correlations\\ Tile\ edge\ discontinuities\\ Accentuated\ spike\ noise\end{cases}$
Motion compensation Havran et al. 2003	*Cam.* *Obj.*	*PAP*	*LP* *EP*	$L(S\|D)^{+}DE$ $MPL=\infty$	$O(FB)$	$\begin{cases}Bias\\ Gloss\ noise\end{cases}$
Light source reconnection Sbert et al. 2004a	*Lgt.*	*PAP*	*LP*	$LD^{+}E$ $MPL=\infty$	est. $O(N_f N_p \Phi)$	$Pontential$ $temporal\ aliasing$
Light source reconnection Sbert et al. 2004b	*Lgt.*	*PAP* *UI*	*LP*	$(LSD)(S\|D)^{+}E$ $MPL=\begin{cases}\infty & LP\\ 1 & EP\end{cases}$	IB	n/a
Light source reconnection Sbert & Castro 2004	*Lgt.*	*PAP*	*LP* GP	$LD^{+}E$ $MPL=\begin{cases}\infty & LP\\ 1 & GP\end{cases}$	$O(5N_f N_p \Phi)$	n/a
Unbiased motion compensation Méndez-Feliu et al. 2006	*Cam.*	*PAP*	*EP*	$L(S\|D)^{*}E$ $MPL=\infty$	Gb	n/a
Path manipulation algorithm	*Cam.* *Lgt.* *Obj.*	*PAP* *UI*	*LP* *EP*	$L(S\|D)^{*}E$ $LS^{+}DE$ $MPL=\infty$	$O(PMA)$	$If\ collateral\ paths$ $are\ inadequately$ $reconstructed \Longrightarrow bias$
Symbols	*Cam. – camera*; *Obj. – objects*; *Lgt. – lights* *PAP – predefined animation paths*; *UI – user input* *EP – eye subpaths*; *LP – light subpaths*; *GP – gathering subpaths* *MPL – maximum subpath length* N_f *– frames number*; N_p *– patches number*; *SPP – samples per pixel*					
Complexities	$\begin{cases} FB = N_f \cdot (k\% + objects) \quad k \in \mathbb{N}^{+} \\ Sz = 4bytes \cdot Img_{width} \cdot Img_{height} \cdot N_f \\ IB = Sz \cdot LP \\ Gb = Sz \cdot \left(5bytes + 5bytes \cdot SPP \cdot \left(N_f - 1\right) + 6bytes \cdot SPP + 4bytes \cdot SPP \cdot N_f\right) \\ PMA = \varsigma_{\varphi_i} + \lvert\mathcal{M}\rvert \quad \varsigma_{\varphi_i} \mapsto equation\ (5.2) \end{cases}$					

Table 2.1*: Comparison between the path manipulation algorithm & 1st set of temporal methods. Unless otherwise specified, reuse occurs between frames and* (LSD) *indicates a light source with zero surface area & finite solid angle emission (Veach 1998, p. 232).*

Method	Dynamism	Interactivity	Reuse	Transport	Memory footprint	Artefacts
Obscurances & motion compensation Méndez-Feliu & Sbert 2006	*Cam. Lgt.*	*PAP*	*IV*	$Obscurances$	$O(Obs)$	n/a
Real-time instant radiosity Knecht 2009	*Cam. Lgt. Obj. Mat.*	*RT*	*IV*	$(LSD)(S\|D)^+E$ $MPL = \begin{cases} 6 & LP \\ 1 & EP \end{cases}$	RIR	$\begin{cases} Clamping\ bias \\ Weak\ VPL\ disposal\ bias \\ Indirect\ shadows\ loss \\ Temp.alias\ (VPLs < 256) \\ Temp.smoothing\ errors \end{cases}$
Population Monte Carlo energy redistribution Lai 2010	*Cam. Obj.*	*PAP*	*LP EP*	$L(S\|D)^*E$ $L(S^+D\|S^+)^+E$ $MPL = \infty$	*453 – 943 Mbytes per frame*	$\begin{cases} Possible\ temp.artefacts \\ Noisier\ dark\ regions \end{cases}$
Sequential Monte Carlo instant radiosity Hedman et al. 2016	*Cam. Lgt.*	*PAP UI*	*VPLs*	LD^+E $MPL = \begin{cases} 3 & LP \\ 1 & EP \end{cases}$	est. RIR-$like$	$\begin{cases} Clamping\ bias \\ Weak\ VPL\ disposal\ bias \\ Density\ estimation\ bias \\ Energy\ loss\ (clamping) \\ Spatial\ aliasing \\ Flickering\ (very\ low) \end{cases}$
Temporal image-space gradients Manzi et al. 2016	*Cam. Obj.*	*PAP UI*	*RV inter frame EP intra frame*	$L(S\|D)^*E$ $MPL = \infty$	*Orders of Gbytes*	$\begin{cases} Flickering \\ High\text{-}speed\ degradation \end{cases}$
Path manipulation algorithm	*Cam. Lgt. Obj.*	*PAP UI*	*LP EP*	$L(S\|D)^*E$ LS^+DE $MPL = \infty$	$O(PMA)$	$If\ collateral\ paths\ are\ inadequately\ reconstructed \Longrightarrow bias$

Symbols	*Cam. – camera; Lgt. – lights; Obj. – objects; Mat. – materials* *PAP – predefined animation paths; RT – real time; UI – user input* *IV – illumination values; LP – light subpaths; EP – eye subpaths* *VPLs – virtual point lights; RV – random variables* *MPL – maximum subpath length* N_{lf} *– light frames number;* N_{rf} *– reused frames number*
Complexities	$\begin{cases} Obs = Img_{width} \cdot Img_{height} \cdot (1 + N_{lf} + 2 \cdot N_{rf} \cdot (1 + N_{lf})) \\ RIR = \|buffers\| + O(VPLs) + \|imperfect\ shadow\ maps\| \\ PMA = \varsigma_{\varphi_i} + \|\mathcal{M}\| \quad \varsigma_{\varphi_i} \mapsto equation\ (5.2) \end{cases}$

***Table 2.2**: Comparison between the path manipulation algorithm & 2^{nd} set of temporal methods. Unless otherwise specified, reuse occurs between frames and (LSD) indicates a light source with zero surface area & finite solid angle emission (Veach 1998, p. 232).*

2.7 Conclusions

The present chapter discussed the foundation of light transport theory, identified the best algorithm for the development of the light transport framework and analysed the current work in relation with the recent advancements in light transport simulation.

The light transport theory began by defining the solid angle (section 2.1) and the radiometric quantities (section 2.2). Surface scattering (section 2.3) introduced the bidirectional scattering distribution function (subsection 2.3.1), with its specialization (subsection 2.3.2), as the main mathematical tools that describe the local behaviour of light. The scattering equation (subsection 2.3.4) served to derive the rendering equation. The analytic solution of the rendering equation (subsection 2.4.1) emphasized its properties, whereas the surface domain (subsection 2.4.2) and the path integral (subsection 2.4.3) formulations provided two different approaches to interpreting it.

The review of the global illumination classes (section 2.5), identified the algorithms (subsection 2.5.8) that fulfilled the prerequisite established for the development of the light transport framework. Bidirectional path tracing (subsection 2.5.4.2) prevailed as the best development foundation, due to its robustness, lack of bias and simplicity. However, the goal of the current work is to simulate high-quality illumination effects in conditions of dynamic geometry. Consequently, the current work ports bidirectional path tracing to the temporal domain via an apparatus of sampling and reuse strategies. The path manipulation algorithm reconstructs and reuses paths across frames, with the result that the static path manipulation is replaced with a generation-evaluation-reuse cycle. The temporal aware approach circumvents the exhaustive regeneration of paths, reduces the computational load of the original algorithm and supports scene dynamism.

The temporal aware manipulation of paths is the major difference between the path manipulation algorithm and the recent advances that use sampling oriented techniques to improve the efficiency of the light transport simulation (subsection 2.6.1). Unlike these algorithms, the proposed algorithm does not regenerate the entire path collection between iterations, instead it reconstructs and reuses paths on a frame-to-frame basis.

The implementation of the path manipulation algorithm (chapter 4) incorporates techniques developed in progressive Monte Carlo ray tracing (subsection 2.6.2). However, the focus is not on adapting bidirectional path tracing to the rigours of the stream processing technology. The aim is to support scene dynamism by explicitly exploiting the temporal coherence of illumination, through the reconstruction and reuse of entire light transport paths. Accelerated classic ray tracing (subsection 2.6.2) exploits temporal coherence, but only by refitting the acceleration structures and/or tracing packets of rays. Moreover, it does not compute the radiance equilibrium of a scene, as it traces only primary and secondary rays. The path manipulation algorithm exploits the temporal coherence of the paths used to estimate the rendering equation.

Path reuse algorithms (subsection 2.6.3) accommodate scene dynamism by recycling path information. However, they restrict dynamism to certain objects and/or require predefined animation paths. Allowing only the camera or the light sources to move, implies that only the eye or light subpaths will be reused. The techniques that require predefined animation paths are precluded from being used in interactive applications. The path manipulation algorithm eliminates both limitations, by reconstructing and reusing subpaths generically and without the need for predefined animation paths. Any scene object can be transformed and the subpaths are reconstructed regardless of their type. As long as the objects and transformations can be deduced from the user's actions, the path manipulation algorithm can also reconstruct subpaths based on the user input. The proposed algorithm equally supports user input and predefined animation paths, with the result that its applicability ranges from animation to interaction based systems. For example, the path manipulation algorithm can be used with CAD/CAM systems to design a commercial product or render a recorded assembling sequence. Chapter 5 further details its applicability scope, by analysing its results in various scenarios. The next chapter delineates the theoretical foundation of the path manipulation algorithm.

Chapter 3

Path manipulation strategies

The essential contribution of the current work is the extension of bidirectional path tracing towards reusing paths in the temporal domain. To support the generation of a wide range of light phenomena, in conditions of dynamic geometry, path manipulation strategies reconstruct and reuse light transports paths across different rendering frames.

This chapter defines path manipulation from a mathematical standpoint. Relying on the original formulation (Veach 1998, p. 302-307), the first section examines the local path sampling techniques and the sample contribution evaluation that underlie the bidirectional path tracing algorithm. The purpose of this first section is to establish the standard framework in which path sampling techniques operate and set the foundation for the mathematical description of the path manipulation algorithm. The next sections discuss the original contributions of this work and demonstrate that the path sampling techniques can be used to extend the path lifespan from a generation-evaluation cycle to a generation-evaluation-reuse one. Path sampling techniques are used in conjunction with path reuse to define an apparatus for path manipulation, capable of supporting dynamism in Monte Carlo light transport simulations. Regarding dynamism, the path manipulation algorithm addresses only the geometric transformations of the scene. It is presupposed that the emission and the scattering models are maintained unaltered.

Section 3.2 defines the concepts of path validity and immutable contribution in the new generation-evaluation-reuse context. Section 3.3 analyses the first step of the path manipulation algorithm, which identifies the invalid paths and computes their anchors. Section 3.4 discusses the reconstruction of invalid paths and the evaluation of their contributions. As the second algorithmic step, reconstruction subsumes the primary anchor, two-chain and terminus anchor scenarios. The two-chain reconstruction covers the novel intra-subpath connectivity strategy. Section 3.5 gives a high-level description of the path manipulation algorithm, whereas section 3.6 carries a formal analysis on it.

3.1 Background: sampling and estimate computation

Chapter 2 (subsection 2.5.4.2) briefly discussed the sampling and evaluation of paths in bidirectional path tracing. This section inspects the sampling techniques as the prime mechanism for path generation and emphasizes their role in the evaluation of the path contributions. A full mathematical derivation, with examples, identifies and computes the factors that define the contributions of the sampled paths. The goal of this analysis is to establish the primary concepts on which the path manipulation algorithm operates.

3.1.1 Local path sampling techniques

Bidirectional path tracing can generate a variety of illumination effects for an extended range of complex lighting, scattering and geometric models. Its robustness stems from optimally combining paths generated with different local sampling techniques into low-variance estimators. The local path sampling techniques and multiple importance sampling are the fundamental mechanisms that underlie bidirectional path tracing.

The local path sampling techniques construct paths incrementally, based on the local information of the last sampled vertices. Three basic sampling methods are used to construct paths (Veach 1998, p. 226). The first method samples a vertex according to a predefined distribution over the scene surfaces. For instance, vertices on non-delta light sources are sampled according to the light source emissivity. Vertices on lenses are also sampled with this method. The second method uses a local probability density function to sample a direction that would produce the next path vertex by intersecting the closest surface. Most vertices are generated by sampling preceding BSDFs. The third method creates a complete path by connecting the visible vertices of two subpaths.

The local path sampling techniques can generate paths arbitrarily from the light source, the camera or other scene surfaces. Bidirectional path tracing generates light transport paths by connecting a subpath sampled from the light to another sampled from the eye.

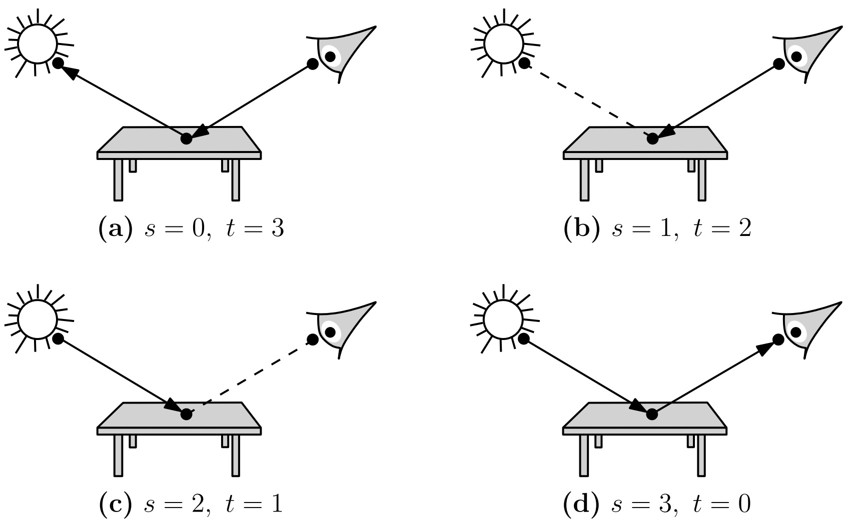


Figure 3.1*: Paths of length $k = 2$ may be sampled by (a) naïve path tracing, (b) path tracing with next event estimation, (c) tracing photons until they intersect surfaces visible from the camera or (d) tracing photons up to the camera (Veach 1998, p. 299).*

The path integral framework defines a measurement associated with a given pixel j as:

$$I_j = \int_{\mathfrak{X}} f_j(\bar{x}) d\mu(\bar{x}) \tag{3.1}$$

where $\bar{x} = x_0 \ldots x_k$ is a path of length k, $d\mu(\bar{x}) = dA(x_0) \ldots dA(x_k)$ is the differential area-product measure and $f_j(\bar{x})$ is the measurement contribution function defined as:

$$\begin{aligned} f_j(\bar{x}) \;=\; & L_e(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \\ & \prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1})\, W_e^j(x_{k-1} \to x_k) \end{aligned} \tag{3.2}$$

The path $\bar{x} = x_0 \ldots x_k$ is sampled according to a probability density $p_{s,t}$. Sampling from $p_{s,t}$ means generating a complete light transport path by connecting a light subpath of random length s and vertices $y_0 \ldots y_{s-1}$ to an eye subpath of random length t and vertices $z_{t-1} \ldots z_0$. The bidirectional path has the form $\bar{x} \equiv \bar{x}_{s,t} = y_0 \ldots y_{s-1} z_{t-1} \ldots z_0$. The connecting edge $y_{s-1} z_{t-1}$ establishes the connection between the two subpaths.

Bidirectional path tracing estimates integral (3.1) by drawing samples from different such probability density functions. An entire family of sampling techniques is obtained by varying the number of vertices in both the light and the eye subpath. For a path of length $k = s + t - 1$, there are $k + 2$ different sampling techniques that could generate the path. Figure 3.1 depicts the 4 techniques that could sample a path of length $k = 2$.

Likewise, paths of length $k = 5$ can be sampled through any of the seven techniques:

ST_1:	$s = 0, t = 6$	$[x_0x_1x_2x_3x_4x_5]^E$
ST_2:	$s = 1, t = 5$	$[x_0]^L[x_1x_2x_3x_4x_5]^E$
ST_3:	$s = 2, t = 4$	$[x_0x_1]^L[x_2x_3x_4x_5]^E$
ST_4:	$s = 3, t = 3$	$[x_0x_1x_2]^L[x_3x_4x_5]^E$
ST_5:	$s = 4, t = 2$	$[x_0x_1x_2x_3]^L[x_4x_5]^E$
ST_6:	$s = 5, t = 1$	$[x_0x_1x_2x_3x_4]^L[x_5]^E$
ST_7:	$s = 6, t = 0$	$[x_0x_1x_2x_3x_4x_5]^L$

where $ST_{1\leq i\leq 7}$ is the i^{th} sampling technique and E/L identifies an eye or a light vertex.

Each sampling technique corresponds to a different probability density $p_{s,t}$ over the space of paths. This means that each technique samples different factors from the measurement contribution function (3.2) and thus accounts for different illumination effects. Bidirectional path tracing generates samples using all these techniques and combines them in low-variance estimators through multiple importance sampling, i.e.

$$F = \sum_{s\geq 0}\sum_{t\geq 0} w_{s,t}(\bar{x}_{s,t}) \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} \tag{3.3}$$

where $w_{s,t}$ is the weighting function computed through a sample combination strategy. Veach and Guibas (1995) propose several heuristics that optimally combine samples.

3.1.2 Evaluation of the sample contribution

A sample is a light transport path $\bar{x}_{s,t}$ generated according to the probability density $p_{s,t}$ through the local path sampling techniques described in the previous subsection.

The contribution of a sample $\bar{x}_{s,t}$ can be defined using the multi-sample estimator (3.3):

$$C_{s,t} = w_{s,t}(\bar{x}_{s,t}) \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} \tag{3.4}$$

Given $\bar{x}_{s,t}$ is composed of independent subpaths, its contribution $C_{s,t}$ can be factored into terms that depend either on one of the subpaths or on the connecting edge $y_{s-1}z_{t-1}$:

$$C_{s,t} = w_{s,t}\eta_s c_{s,t}\eta_t \tag{3.5}$$

where η_s is the factor dependent on the light subpath, η_t is the factor dependent on the eye subpath and $c_{s,t}$ is the factor dependent solely on the connecting edge $y_{s-1}z_{t-1}$.

The independent throughput of the light subpath η_s can be computed recursively, based exclusively on the constituent s vertices of the subpath, as demonstrated below:

$$\eta_0 = 1$$

$$\eta_1 = \frac{L_e(y_0)}{P_A(y_0)}$$

$$\eta_2 = \frac{L_e(y_0 \to y_1)}{P^{\perp}(y_0 \to y_1)}\eta_1$$

$$\eta_i = \frac{f_s(y_{i-3} \to y_{i-2} \to y_{i-1})}{P^{\perp}(y_{i-2} \to y_{i-1})}\eta_{i-1} \qquad 3 \le i \le s \tag{3.6}$$

where L_e is the radiance emitted from a light source, P_A is the probability of the first vertex y_0 measured with respect to the light surface area and $P^{\perp}$ is the probability of any subsequent light subpath vertex, measured with respect to the projected solid angle.

Formulae (3.6) exhibit a difference in measure between the probabilities of the first and the next vertices of the light subpath. This apparent inconsistency is justified by the fact that the geometrical factor $G(y_{i-1} \leftrightarrow y_i)$ appears in the nominator of each $\eta_{i+1|i\geq 1}$, in accordance with the definition of the measurement contribution function. Also, the probabilities measured with respect to the surface area can be expressed as:

$$P_A(y_i) = P^{\perp}(y_{i-1} \to y_i)G(y_{i-1} \leftrightarrow y_i) \qquad \forall i \geq 1 \tag{3.7}$$

Therefore, the geometrical factors cancel each other out yielding the formulae in (3.6).

The throughput of the eye subpath η_t can be computed using the following formulae:

$$\begin{aligned}
\eta_0 &= 1 \\
\eta_1 &= \frac{W_e(z_0)}{P_A(z_0)} \\
\eta_2 &= \frac{W_e(z_1 \to z_0)}{P^{\perp}(z_0 \to z_1)}\eta_1 \\
\eta_i &= \frac{f_s(z_{i-1} \to z_{i-2} \to z_{i-3})}{P^{\perp}(z_{i-2} \to z_{i-1})}\eta_{i-1} \qquad 3 \leq i \leq t
\end{aligned} \tag{3.8}$$

where W_e is the responsivity of the sensor to incoming light and f_s is the adjoint BSDF.

The $c_{s,t}$ term subsumes the residual factors of the measurement contribution function and depends uniquely on the scattering that occurs along the connecting edge $y_{s-1}z_{t-1}$:

$$c_{s,t} = f_s(y_{s-2} \to y_{s-1} \to z_{t-1})G(y_{s-1} \leftrightarrow z_{t-1})f_s(y_{s-1} \to z_{t-1} \to z_{t-2}) \quad \forall s,t > 0 \tag{3.9}$$

The BSDF is substituted by $L_e(y_0 \to z_{t-1})$ for $s = 1$ and by $W_e(y_{s-1} \to z_0)$ for $t = 1$. When $s = 0$ or $t = 0$, then $c_{0,t} = L_e(z_{t-1} \to z_{t-2})$ respectively $c_{s,0} = W_e(y_{s-2} \to y_{s-1})$.

Formulae (3.6), (3.8) and (3.9) determine the unweighted contribution of a sample $\bar{x}_{s,t}$:

$$C^*_{s,t} \equiv \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} = \eta_s c_{s,t} \eta_t \tag{3.10}$$

where $p_{s,t}(\bar{x}_{s,t})$ is the probability density associated with the entire light transport path.

The probability density $p_{s,t}(\bar{x}_{s,t})$ can be computed as the product between the probability density of the light subpath p_s and the probability density of the eye subpath p_t, that is

$$p_{s,t}(\bar{x}_{s,t}) = p_s p_t \tag{3.11}$$

The probability density of one subpath is independent of the other subpath and can be computed as the product of all the probability densities with which the subpath vertices were sampled. That is, both p_s and p_t can be computed recursively as indicated below:

$$\begin{aligned} &p_0 = 1 \\ &p_1 = P_A(x_0) \\ &p_i = P^{\perp}(x_{i-2} \rightarrow x_{i-1}) G(x_{i-2} \leftrightarrow x_{i-1}) p_{i-1} \quad 2 \leq i \leq \{s|t\} \end{aligned} \tag{3.12}$$

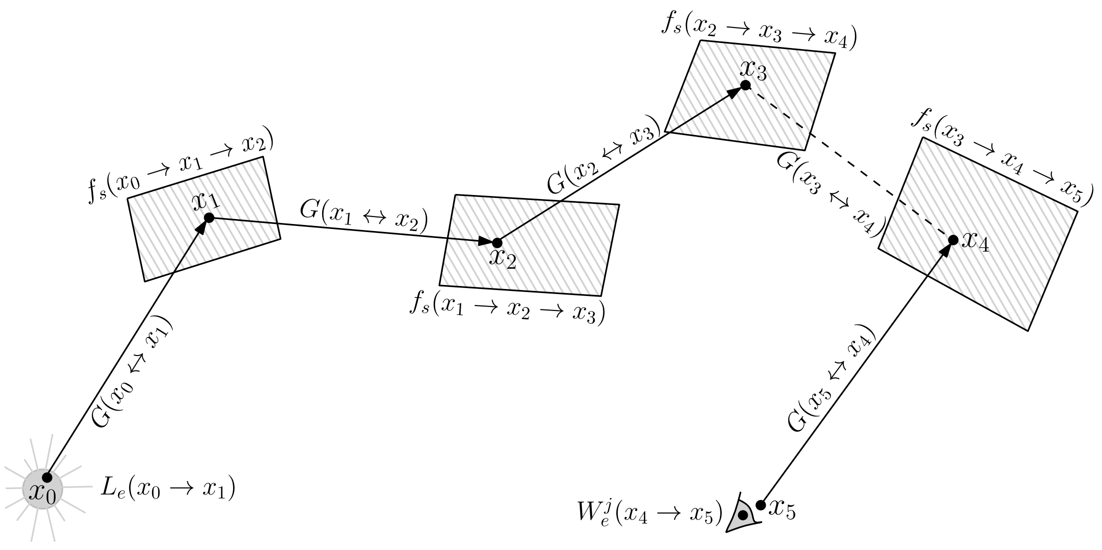
where x_i is a generic reference to a vertex belonging to either a light or an eye subpath.

To have a complete evaluation of the sample contribution (3.4), the weighting function $w_{s,t}$ must be computed. As opposed to the unweighted contribution $C^*_{s,t}$, which depends uniquely on the information generated by sampling the path $\bar{x}_{s,t}$ from the density $p_{s,t}$, the weighting function $w_{s,t}$ depends on all the probability densities with which $\bar{x}_{s,t}$ may be generated by the $k+2$ sampling techniques. Hence, $s+t+1$ probability densities are required to determine $w_{s,t}$. Let these probability densities be denoted by:

$$p_i \equiv p_{i,k+1-i} \qquad\qquad 0 \leq i \leq k+1 \tag{3.13}$$

where $p_s = p_{s,t}$ is the probability density associated with the actual generation of $\bar{x}_{s,t}$.

The actual probability density p_s suffices to compute the other $k+1$ probability densities $p_0 \dots p_{s-1}, p_{s+1} \dots p_{k+1}$, as each of them can be defined relative to p_s. Consider a path of length $k = 5$, generated with a probability density $p_4 = p_{4,2}$ as in Figure 3.2.



$f_s(x_2 \to x_3 \to x_4)$
x_3
$f_s(x_3 \to x_4 \to x_5)$
$G(x_2 \leftrightarrow x_3)$
$f_s(x_0 \to x_1 \to x_2)$
x_1
$G(x_1 \leftrightarrow x_2)$
$G(x_3 \leftrightarrow x_4)$
x_4
x_2
$f_s(x_1 \to x_2 \to x_3)$
$G(x_0 \leftrightarrow x_1)$
$G(x_5 \leftrightarrow x_4)$
x_0
$L_e(x_0 \to x_1)$
$W_e^j(x_4 \to x_5)$
x_5

***Figure 3.2**: Scattering events along a path of length $k = 5$, sampled with a probability density $p_{4,2}$. Vertices $x_0 - x_3$ compose the light subpath and $x_4 - x_5$ the eye subpath.*

As exemplified in subsection 3.1.1, a path of length $k = 5$ can be sampled through $k + 2 = 7$ different techniques, each with its own probability density $p_{0 \leq i \leq k+1}$. Had the path been generated using the i^{th} sampling technique, then the probability density p_i could be computed via equation (3.11). This assumption yields the ensuing densities:

$$ST_1: \quad s = 0, t = 6 \quad p_0 = p_{0,6} = [P_A(x_0)P_A(x_1)P_A(x_2)P_A(x_3)P_A(x_4)P_A(x_5)]^E$$

$$ST_2: \quad s = 1, t = 5 \quad p_1 = p_{1,5} = [P_A(x_0)]^L[P_A(x_1)P_A(x_2)P_A(x_3)P_A(x_4)P_A(x_5)]^E$$

$$ST_3: \quad s = 2, t = 4 \quad p_2 = p_{2,4} = [P_A(x_0)P_A(x_1)]^L[P_A(x_2)P_A(x_3)P_A(x_4)P_A(x_5)]^E$$

$$ST_4: \quad s = 3, t = 3 \quad p_3 = p_{3,3} = [P_A(x_0)P_A(x_1)P_A(x_2)]^L[P_A(x_3)P_A(x_4)P_A(x_5)]^E$$

$$ST_5: \quad s = 4, t = 2 \quad p_4 = p_{4,2} = [P_A(x_0)P_A(x_1)P_A(x_2)P_A(x_3)]^L[P_A(x_4)P_A(x_5)]^E$$

$$ST_6: \quad s = 5, t = 1 \quad p_5 = p_{5,1} = [P_A(x_0)P_A(x_1)P_A(x_2)P_A(x_3)P_A(x_4)]^L[P_A(x_5)]^E$$

$$ST_7: \quad s = 6, t = 0 \quad p_6 = p_{6,0} = [P_A(x_0)P_A(x_1)P_A(x_2)P_A(x_3)P_A(x_4)P_A(x_5)]^L$$

where $ST_{1 \leq i \leq 7}$ denotes the i^{th} sampling technique, $x_{0 \leq j \leq 5}$ refers to either a light (L) or an eye (E) subpath vertex and $P_A(x_j)$ is the surface area probability associated with x_j.

However, the considered path was assumed to be generated with a probability density $p_4 = p_{4,2}$. The probability densities $p_0 - p_3$ and $p_5 - p_6$ represent the other $k+1$ ways in which the path could have been sampled and they must to be computed relative to p_4. To do so, means to compute the ratio between adjacent probability densities, i.e.

$$\frac{p_1}{p_0} = \frac{[P_A(x_0)]^L}{[P_A(x_0)]^E} = \frac{P_A(x_0)}{P^\perp(x_1 \to x_0)G(x_0 \leftrightarrow x_1)}$$

$$\frac{p_2}{p_1} = \frac{[P_A(x_1)]^L}{[P_A(x_1)]^E} = \frac{P^\perp(x_0 \to x_1)G(x_0 \leftrightarrow x_1)}{P^\perp(x_2 \to x_1)G(x_1 \leftrightarrow x_2)}$$

$$\vdots$$

$$\frac{p_6}{p_5} = \frac{[P_A(x_5)]^L}{[P_A(x_5)]^E} = \frac{P^\perp(x_4 \to x_5)G(x_4 \leftrightarrow x_5)}{P_A(x_5)}$$

where equality (3.7) was used to expand the probabilities measured with respect to the surface area. Expressing the probability densities with respect to the projected solid angle evinces the directions from which the vertices $x_{0\leq j\leq 5}$, found in the nominator and the denominator of each fraction, are assumed to have been generated. Relative to how it was sampled, a vertex can be processed either in a forward or a reverse direction.

The above probability density ratios can be generalised to paths of arbitrary length k:

$$\frac{p_1}{p_0} = \frac{[P_A(x_0)]^L}{[P_A(x_0)]^E} = \frac{P_A(x_0)}{P^\perp(x_1 \to x_0)G(x_0 \leftrightarrow x_1)}$$

$$\frac{p_{i+1}}{p_i} = \frac{[P_A(x_i)]^L}{[P_A(x_i)]^E} = \frac{P^\perp(x_{i-1} \to x_i)G(x_{i-1} \leftrightarrow x_i)}{P^\perp(x_{i+1} \to x_i)G(x_i \leftrightarrow x_{i+1})} \qquad 0 < i < k$$

$$\frac{p_{k+1}}{p_k} = \frac{[P_A(x_k)]^L}{[P_A(x_k)]^E} = \frac{P^\perp(x_{k-1} \to x_k)G(x_{k-1} \leftrightarrow x_k)}{P_A(x_k)} \qquad (3.14)$$

The other $k+1$ probability densities can be easily computed by using the probability density $p_s = p_{s,t}$ and the ratios defined in (3.14). Starting from p_s, the probability densities $p_{s+1} \dots p_{k+1}$ can be computed recursively using the last two ratios in (3.14). The densities $p_0 \dots p_{s-1}$ can be derived from the first ratio and the reciprocal p_i/p_{i+1}.

Based on the probability densities $p_{0 \le i \le k+1}$, the weighting function $w_{s,t}$ can finally be computed through any low-variance combination strategy (Veach and Guibas 1995). By using the power heuristic with an exponent $\beta = 2$, the weights can be computed as:

$$w_{s,t}(\bar{x}_{s,t}) = \frac{p_s^2}{\sum_i p_i^2} \tag{3.15}$$

The evaluation of the weights can be optimized by exploiting the relative dependence of the probability densities $p_0 \dots p_{s-1}, p_{s+1} \dots p_{k+1}$ on the actual probability density p_s. The latter can be factored out of the sum $\sum_i p_i^2$ and used to simplify the fraction (3.15):

$$w_{s,t}(\bar{x}_{s,t}) = \frac{p_s^2}{\sum_i p_i^2} = \frac{1}{\sum_i (p_i/p_s)^2} \tag{3.16}$$

As remarked by Veach (1998), the weighting function (3.16) depends solely on the probability density ratios (3.14). The fraction p_i/p_s refers to a product of probability density ratios without the p_s factor and does not require the computation of $p_{0 \le i \le k+1}$.

For the path of length $k = 5$, the actual density $p_4 = p_{4,2}$ can be factored out as shown:

$$\begin{aligned}
\sum_{i=0}^{6} p_i \quad = \quad & \underbrace{p_4 \frac{[P_A(x_3)P_A(x_2)P_A(x_1)P_A(x_0)]^E}{[P_A(x_3)P_A(x_2)P_A(x_1)P_A(x_0)]^L}}_{p_0} + \underbrace{p_4 \frac{[P_A(x_3)P_A(x_2)P_A(x_1)]^E}{[P_A(x_3)P_A(x_2)P_A(x_1)]^L}}_{p_1} + \\
& \underbrace{p_4 \frac{[P_A(x_3)P_A(x_2)]^E}{[P_A(x_3)P_A(x_2)]^L}}_{p_2} + \underbrace{p_4 \frac{[P_A(x_3)]^E}{[P_A(x_3)]^L}}_{p_3} + \\
& p_4 + \\
& \underbrace{p_4 \frac{[P_A(x_4)]^L}{[P_A(x_4)]^E}}_{p_5} + \underbrace{p_4 \frac{[P_A(x_4)P_A(x_5)]^L}{[P_A(x_4)P_A(x_5)]^E}}_{p_6}
\end{aligned}$$

Finally, the weighted contribution of the path $\bar{x}_{s,t}$ can be completely determined using:

$$C_{s,t} = w_{s,t} C_{s,t}^* = w_{s,t} \eta_s c_{s,t} \eta_t$$

Such contributions are accrued through the multi-sample estimator $F = \sum_{s \ge 0} \sum_{t \ge 0} C_{s,t}$.

3.2 Path validity and immutable contribution

So far, path sampling has been analysed strictly as the fundamental tool underlying the versatility of bidirectional path tracing. It was essential to evince the role sampling techniques play in path generation and to peruse their interpretability and applicability with respect to the estimation of the measurement equation. Yet, for the set goal such an analysis would be incomplete if restricted to the generation and evaluation of paths. The reason for such incompleteness stands precisely in the fact that the path lifespan is limited to the generation-evaluation cycle. Paths are simply discarded after the evaluation of their contributions and the next frame is produced via a new generation-evaluation cycle. Such a cycle imposes a static usage of paths and sampling techniques.

To inject reuse within the generation-evaluation cycle of a path, it is necessary to redefine that path as an entity that spans different such cycles. Let $\bar{x}_{s,t}^{\varphi_i}$ be a path sampled from a probability density $p_{s,t}$ that brings the standard contribution to the multi-sample estimator (3.3). Let its contribution be assessed for the current frame φ_i:

$$C_{s,t}(\bar{x}_{s,t}^{\varphi_i}) = w_{s,t}\left(\bar{x}_{s,t}^{\varphi_i}\right)\frac{f_j\left(\bar{x}_{s,t}^{\varphi_i}\right)}{p_{s,t}\left(\bar{x}_{s,t}^{\varphi_i}\right)}$$

The above contribution remains valid for all the frames $\varphi_{i+1} \dots \varphi_{k-1}$ that were not altered via geometric transformations. If scene dynamism occurs between frames φ_{k-1} and φ_k, then the validity of $\bar{x}_{s,t}^{\varphi_i}$ must be determined relative to the configuration existent in φ_k. Figure 3.3 exemplifies how the transformation of two scene objects affects the validity of the paths between the given frame φ_{k-1} and its successor φ_k.

Two basic phenomena define the consistency of a path $\bar{x}_{s,t}^{\varphi_i}$ relative to a new frame φ_k. Both underlie the measurement contribution function (3.2) and regard the propagation and the scattering events. Light propagation is described via the geometric operator:

$$L_i(x_i,\vec{\omega}_i) = (\mathcal{P}L_o)(x_i,\vec{\omega}_i) = \begin{cases} L_o(\tau(x_i,\vec{\omega}_i),-\vec{\omega}_i) & if \ \ \alpha_{min}(x_i,\vec{\omega}_i) < \infty \\ 0 & otherwise \end{cases} \qquad (3.17)$$

where x_i is a path vertex, $\alpha_{min}(x_i,\vec{\omega}_i)$ is the boundary distance function and $\tau(x_i,\vec{\omega}_i)$ is the tracing function, which computes the first intersection along the ray $(x_i,\vec{\omega}_i)$. The geometric operator assesses radiance and importance indistinctively: $W_i = \mathcal{P}W_o$.

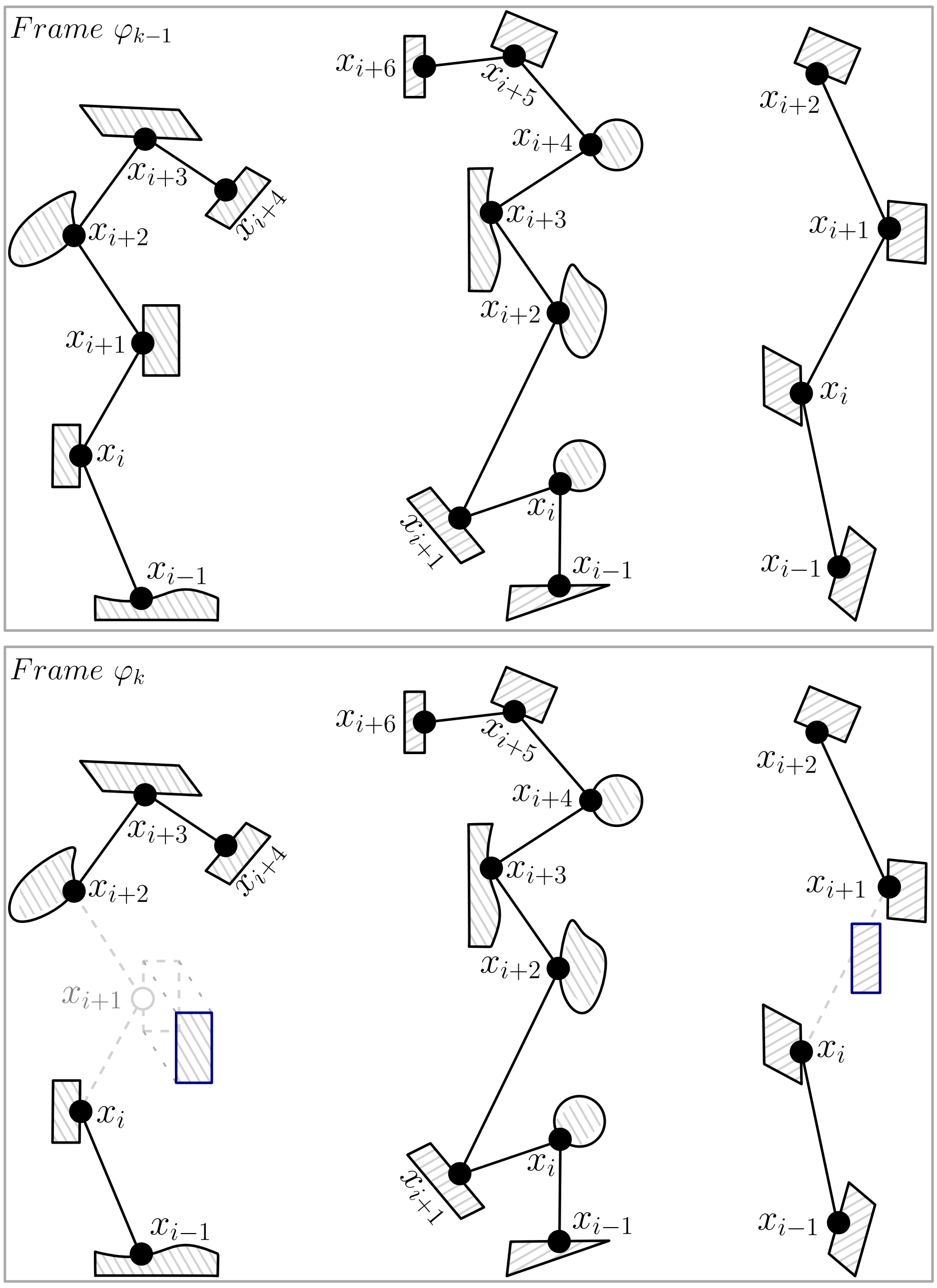


Figure 3.3*: The geometric transformation of the scene between frames φ_{k-1} and φ_k. Dynamic objects (navy) disrupt the left & right paths, without affecting the central path.*

The first condition for a valid contribution $\left[f_j\left(\bar{x}_{s,t}^{\varphi_i}\right)\right]^{\varphi_k}$ is that the tracing function yield, on each segment of $\bar{x}_{s,t}^{\varphi_i}$, the same intersection that was obtained during path generation. That is, each vertex x_i must retain its successor for the traced direction $\vec{\omega}_i = \widehat{x_{i+1} - x_i}$:

$$[\tau(x_i, \vec{\omega}_i)]^{\varphi_k} = x_{i+1} \qquad 0 \le i < s + t - 1 \tag{3.18}$$

Condition (3.18) implies unobstructed visibility between any pair of vertices $x_i - x_{i+1}$:

$$[V(x_i \leftrightarrow x_{i+1})]^{\varphi_k} = 1 \qquad 0 \le i < s + t - 1 \tag{3.19}$$

If condition (3.19) holds, then the geometric operator preserves the propagation events, under the assumption that the emission and the scattering models remain unmodified.

The second condition for $\left[f_j\left(\bar{x}_{s,t}^{\varphi_i}\right)\right]^{\varphi_k}$ to be valid, is that the scattering along $\bar{x}_{s,t}^{\varphi_i}$ be preserved as well. This requirement is ensured by the conservation of the propagation events and by the invariability of the emission and scattering models. The preservation of the scattering events, due to invariable propagation, can be demonstrated as follows:

$$\begin{array}{ccc}
\underbrace{[\mathcal{P}L_e(x_0,\vec{\omega}_0)]^{\varphi_k}}_{[L_i(x_1,-\vec{\omega}_0)]^{\varphi_k}} & & \underbrace{[f_s(x_1,-\vec{\omega}_0,\vec{\omega}_1)\mathcal{P}L_e(x_0,\vec{\omega}_0)]^{\varphi_k}}_{[L_o(x_1,\vec{\omega}_1)]^{\varphi_k}} \\
\| & \Rightarrow & \| \\
\underbrace{[\mathcal{P}L_e(x_0,\vec{\omega}_0)]^{\varphi_i}}_{[L_i(x_1,-\vec{\omega}_0)]^{\varphi_i}} & & \underbrace{[f_s(x_1,-\vec{\omega}_0,\vec{\omega}_1)\mathcal{P}L_e(x_0,\vec{\omega}_0)]^{\varphi_i}}_{[L_o(x_1,\vec{\omega}_1)]^{\varphi_i}} \\
 & \vdots & \\
\underbrace{[\mathcal{P}L_o(x_{s+t-3},\vec{\omega}_{s+t-3})]^{\varphi_k}}_{[L_i(x_{s+t-2},-\vec{\omega}_{s+t-3})]^{\varphi_k}} & & \underbrace{[f_s(x_{s+t-2},-\vec{\omega}_{s+t-3},\vec{\omega}_{s+t-2})\mathcal{P}L_o(x_{s+t-3},\vec{\omega}_{s+t-3})]^{\varphi_k}}_{[L_o(x_{s+t-2},\vec{\omega}_{s+t-2})]^{\varphi_k}} \\
\| & \Rightarrow & \| \\
\underbrace{[\mathcal{P}L_o(x_{s+t-3},\vec{\omega}_{s+t-3})]^{\varphi_i}}_{[L_i(x_{s+t-2},-\vec{\omega}_{s+t-3})]^{\varphi_i}} & & \underbrace{[f_s(x_{s+t-2},-\vec{\omega}_{s+t-3},\vec{\omega}_{s+t-2})\mathcal{P}L_o(x_{s+t-3},\vec{\omega}_{s+t-3})]^{\varphi_i}}_{[L_o(x_{s+t-2},\vec{\omega}_{s+t-2})]^{\varphi_i}}
\end{array} \quad (3.20)$$

where L_e is the light source radiance emitted from x_0, L_i is the radiance incident at a given vertex, L_o is the radiance scattered from a given vertex, $\vec{\omega}_i = \widehat{x_{i+1} - x_i}, \forall i \geq 0$ and $f_s\mathcal{P}$ represents a single scattering step along the path $\bar{x}_{s,t}^{\varphi_i}$. In fact, $f_s\mathcal{P}$ is a simplified version of the light transport operator $T = K\mathcal{P}$, where K is the scattering operator defined as $L_o(x_i,\vec{\omega}_o) = (KL_i)(x_i,\vec{\omega}_o) = \int_{\mathcal{D}} f_s(x_i,\vec{\omega}_i,\vec{\omega}_o)\, L_i(x_i,\vec{\omega}_i)\left|\vec{N}_{x_i}\cdot\vec{\omega}_i\right| d\vec{\omega}_i$.

If scattering is preserved, then equations (3.6) and (3.8) hold. Note that condition (3.19) also implies unobstructed visibility along the connecting edge $y_{s-1}z_{t-1}$ and thus a valid $c_{s,t}$ term (3.9). These norms secure an immutable contribution $f_j\left(\bar{x}_{s,t}^{\varphi_i}\right)$ for the frame φ_k:

$$f_j\left(\bar{x}_{s,t}^{\varphi_k}\right) = \left[f_j\left(\bar{x}_{s,t}^{\varphi_i}\right)\right]^{\varphi_k} = f_j\left(\bar{x}_{s,t}^{\varphi_i}\right) \quad (3.21)$$

An immutable contribution means that the path $\bar{x}_{s,t}^{\varphi_i}$ is valid relative to the new scene configuration and is therefore immediately reusable. The lifespan of such a path can be extended from a generation-evaluation cycle to a generation-evaluation-reuse one.

3.3 Invalidation and anchor computation

The previous section discussed the validity and the contribution of the paths unaltered by the geometric transformations of the scene. The current section examines the less trivial case of invalid paths, which cannot be directly reused in subsequent frames. The focus is on delineating the invalidation step of the path manipulation algorithm and on defining one of the primary concepts used in the reconstruction step, namely the anchor.

A path $\bar{x}_{s,t}^{\varphi_i}$ cannot be reused in a frame φ_k if the visibility of a single edge is occluded:

$$\exists x_i, x_{i+1} \quad 0 \leq i < s + t - 1 \quad s.t. \quad [V(x_i \leftrightarrow x_{i+1})]^{\varphi_k} = 0 \tag{3.22}$$

This scenario occurs when a subpath of $\bar{x}_{s,t}^{\varphi_i}$ is disrupted by the ingress of an object inside its scope. Subpaths with occluded edges will be referred to as *in-scope subpaths*.

The second invalidation scenario occurs when a vertex x_i changes its position due the movement of the object on which it resides. This is the reverse of the previous scenario in that a subpath of $\bar{x}_{s,t}^{\varphi_i}$ is disrupted by the egress of a vertex from within its scope. Subpaths with dynamic vertices will be hereafter referred to as *out-of-scope subpaths*.

In both cases, equation (3.21) no longer holds and the path $\bar{x}_{s,t}^{\varphi_i}$ must be reconstructed such that it becomes consistent with the new scene configuration φ_k. The key idea in the reconstruction of an invalid path is to maximize the reuse of existent path information.

Assume for the moment, that the path $\bar{x}_{s,t}^{\varphi_i}$ has been disrupted by a scene object that is neither the camera nor an existing light source. The transformation of the camera or of a light source implies the disruption of a subpath by the movement of its very first vertex and is treated in the discussion of the primary anchor reconstruction scenario.

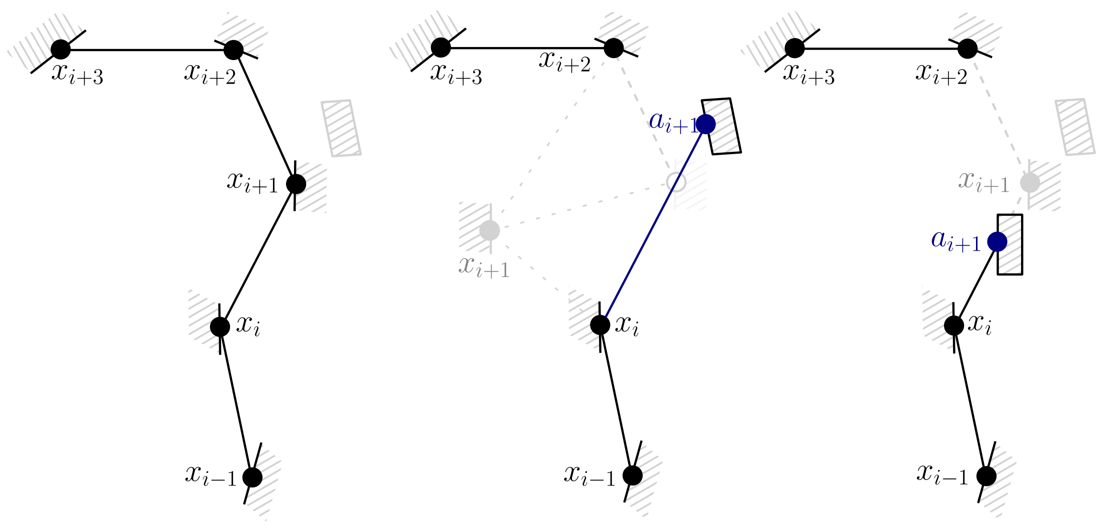


***Figure 3.4**: The in-scope (right) and out-of-scope (centre) invalidations of a path (left). In the out-of-scope invalidation x_{i+1} is positioned at its new location, while a_{i+1} is determined by tracing a ray towards the old position of x_{i+1}. The in-scope invalidation depicts a_{i+1} as the intersection between an occluder and the direction $\vec{\omega}_i = \widehat{x_{i+1} - x_i}$.*

Hence, the first step in reconstructing an invalid path is to determine whether one of its edges was disrupted and if so find the closest intersection along that specific edge. The closest intersection is determined by tracing a ray in the prior direction of the edge. Let $x_i - x_{i+1}$ be a disrupted edge, with $0 \leq i \leq \{s|t\} - 2$. An out-of-scope invalidation is triggered by the movement of the vertex x_{i+1}. In this case, the closest intersection is determined by tracing a ray from x_i towards the old position of x_{i+1}. An in-scope invalidation does not alter the end vertices of an edge, so a ray is always traced in the set direction $\vec{\omega}_i = \widehat{x_{i+1} - x_i}$. In both cases, the closest intersection is determined via:

$$[\tau(x_i, \vec{\omega}_i)]^{\varphi_k} = a_{i+1} \neq x_{i+1} \qquad 0 \leq i \leq \{s|t\} - 2 \tag{3.23}$$

where vertex x_{i+1} assumes the position prior to invalidation whenever x_{i+1} is dynamic. Vertex a_{i+1} is the new successor of x_i and will be referred to as the *anchor*. The index $i + 1$ associated with the anchor is designated as the *anchor level* and it identifies the vertex from which the subpath will be reconstructed. Though the anchor replaces vertex x_{i+1}, it is sampled with the same projected solid angle probability as the latter:

$$P^{\perp}(x_i \to a_{i+1}) = P^{\perp}(x_i \to x_{i+1}) \tag{3.24}$$

The anchor retains the $P^{\perp}$ of the old successor, since the path up to x_i and the direction $\vec{\omega}_i$ are the same for both x_{i+1} and a_{i+1}. Its surface area probability can be computed via (3.7). Figure 3.4 exemplifies the in-scope and the out-of-scope path invalidations.

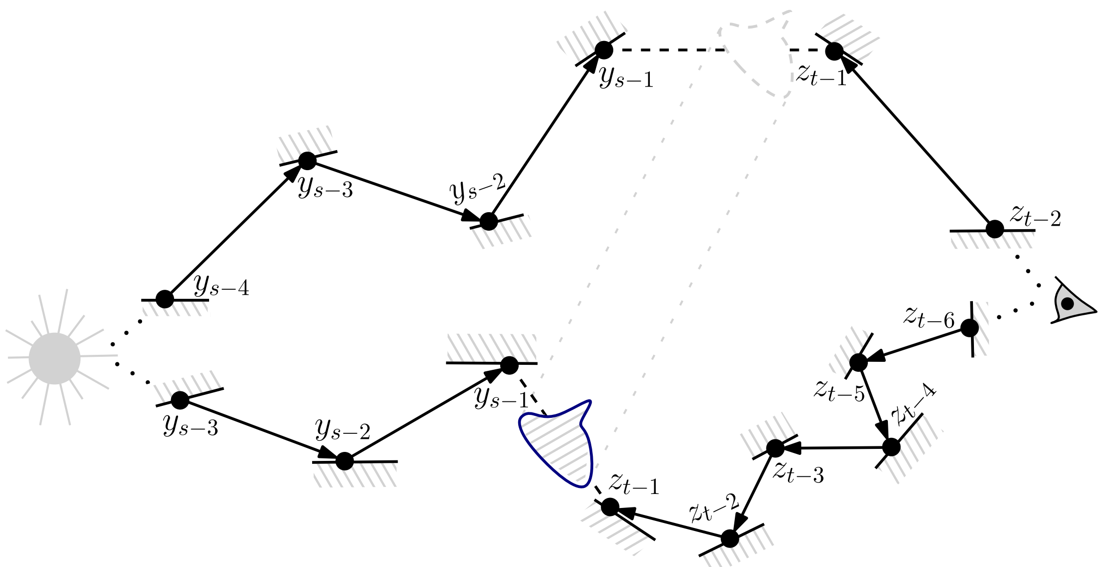


Figure 3.5*: The identification of collateral paths via the connecting edge visibility. Through its movement, the dynamic object (navy) clears the connecting edge of the top path and obstructs that of the bottom path. The top path is processed as collateral to reflect the fact that its constituent subpaths are no longer in the shadow of the dynamic object. The disruption between the subpaths of the bottom path prevents the evaluation of the contribution (3.5) and causes the subpaths to be treated as collateral.*

Condition (3.23) refers only to subpath edges and does not include the connecting edge. There are situations when the subpaths of $\bar{x}_{s,t}^{\varphi_i}$ remain valid, but the connecting edge between them is obstructed. As the geometric structures of the constituent subpaths are valid, new paths can be formed by simply connecting the corresponding subpaths to other homologues. There may also be paths which in the previous frames have been occluded, like those with vertices in previously shadowed areas. The solution is to establish other connections for the given subpaths and re-evaluate their contributions. The paths with geometrically intact subpaths, yet with invalid contributions, will be referred to as *collateral paths*. They can be identified by observing the changes in $V(y_{s-1} \leftrightarrow z_{t-1})$ across frames. Specific approaches to identifying collateral paths are detailed in chapter 4 (subsection 4.3.5). The collateral subpaths can be used to construct new paths, whose contributions can be evaluated with the tools explored in section 3.1. Figure 3.5 depicts two collateral paths identifiable via the connecting edge visibility.

The location of the anchor on a subpath determines three reconstruction scenarios. Together these scenarios describe the second step of the path manipulation algorithm.

3.4 Reconstruction

Reconstruction is the second step of the path manipulation algorithm and its goal is to maximize path reuse by reconnecting disrupted subpath chains into coherent subpaths. The reconstruction of invalid subpaths begins from the anchor and it branches into three scenarios relative to the anchor level. The primary anchor scenario processes out-of-scope subpaths disrupted by the movement of the camera or of a light source. The two-chain scenario uses a novel intra-subpath connectivity strategy to reconnect as many dysfunctional subpath chains as possible. The terminus anchor scenario treats subpaths invalidated along their last edges. All scenarios use the derivation presented in subsection 3.1.2, to delineate the contribution evaluation of the reconstructed paths. Contribution evaluation represents the third step of the path manipulation algorithm.

3.4.1 Primary anchor reconstruction

The current reconstruction scenario processes the out-of-scope subpaths disrupted by the movement of the vertices situated either on a light source or on the camera. Hence, the reconstruction target is any path $\bar{x}_{s,t}^{\varphi_i}$ invalidated by the egress of either x_0 or x_{s+t-1} from within its scope. A dynamic vertex x_0 entails the displacement of the first vertex of a light subpath due to the movement of a light source. Conversely, a dynamic vertex x_{s+t-1} involves the dislocation of the first vertex of an eye subpath by the movement of the camera. The anchor succeeds either vertex and since it is the first point sample seen from a light source or from the camera, it will be referred to as the primary anchor.

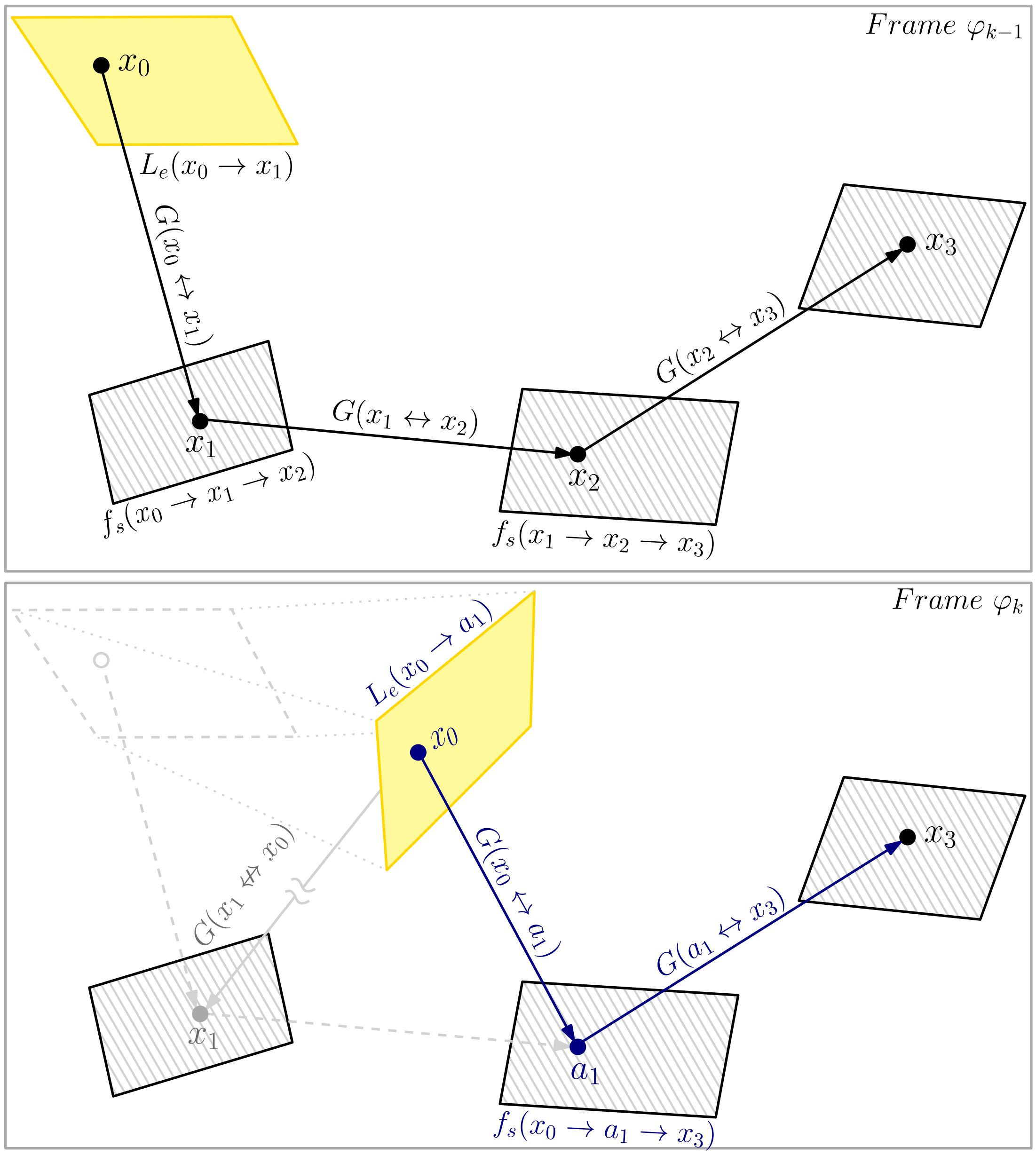
The first step in determining a primary anchor is to sample a new vertex, either on a light source or on the camera lens, using an a priori defined probability distribution over the corresponding surface. Then, a new direction can be generated from this vertex according to the emissivity of the light source or relative to the image plane associated with the camera. The first intersection found along the ray will be the primary anchor.

Once computed, the primary anchor can be reconnected to the rest of the invalid subpath via the intra-subpath connectivity strategy developed for the two-chain reconstruction.

In the case of a failed intra-subpath connectivity attempt, the invalid subpath will be regenerated from the primary anchor by repeatedly sampling the BSDF of the last vertex. Independent connections are established for both the reconstructed subpath and its prior counterpart and their contributions are evaluated according to estimator (3.3).

The primary anchor scenario is a variant of the two-chain reconstruction, as it requires special handling of the primary ray. Unlike scenario (3.23), which entails a change in the end vertex of a traced ray, the primary anchor reconstruction involves a change in the ray origin. This change requires that a new vertex and a new direction be sampled, since no guarantees can be offered regarding the appropriateness of the old direction.

For example, a camera ray usually passes through some region of a pixel. When the camera moves, the old direction may become incongruent with the new position of the camera and of the pixel (e.g. the old direction may not pass through the same pixel). Similarly, a light source may be transformed in such a way that the old direction would actually penetrate its surface or it would altogether emanate from its non-emitting side. Figure 3.6 illustrates the logic of the primary anchor reconstruction for a light subpath.



$Frame\ \varphi_{k-1}$
x_0
$L_e(x_0 \to x_1)$
$G(x_0 \leftrightarrow x_1)$
x_1
$f_s(x_0 \to x_1 \to x_2)$
$G(x_1 \leftrightarrow x_2)$
x_2
$f_s(x_1 \to x_2 \to x_3)$
$G(x_2 \leftrightarrow x_3)$
x_3
$Frame\ \varphi_k$
$L_e(x_0 \to a_1)$
x_0
$G(x_1 \leftrightarrow x_0)$
x_1
$G(x_0 \leftrightarrow a_1)$
a_1
$f_s(x_0 \to a_1 \to x_3)$
$G(a_1 \leftrightarrow x_3)$
x_3

Figure 3.6: *The primary anchor reconstruction of a light subpath (top) triggered by the rotation of an area light source (bottom). A new vertex x_0 is sampled on the light source and the newly generated direction intersects the surface on which the previous vertex x_2 was located. The new anchor a_1 is reconnected to the remaining vertex x_3.*

3.4.2 Two-chain reconstruction

The two-chain reconstruction scenario processes both the in-scope and the out-of-scope paths that are divided in several chains around the anchor a_{i+1}, with $0 \leq i \leq s + t - 3$.

Effectively, an a_{i+1} anchor is located either on a light subpath, on an eye subpath or on the connecting edge between the two. The latter situation regards the collateral paths. As discussed in section 3.3, the solution to solving the disconnection between the two subpaths is to simply reconnect them to other subpaths and to evaluate the contributions of the newly constructed paths with the tools investigated in section 3.1.

When the anchor is located on one of the subpaths, the invalidation affects that subpath alone and does not alter the other subpath or the connecting edge. Therefore, the chains that must be consolidated belong to the same subpath. The unaltered subpath is valid and its throughput is immutable. Consequently, the reconstruction can be performed independently on the altered subpath. This approach logically resembles the evaluation of the sample contribution discussed in subsection 3.1.2. Equation (3.5) decomposes the sample contribution into factors that depend either on just one subpath or on the connecting edge alone. The two-chain reconstruction scenario adopts a similar logic.

3.4.2.1 Path factorization

Analogous to the decomposition of the sample contribution, an invalid path $\bar{x}_{s,t}^{\varphi_i}$ can be divided into three chains, based on the anchor position on the light or the eye subpath:

$$\bar{x}_{s,t}^{\varphi_i} = x_0 \dots x_{s+t-1} = \begin{cases} y_0 \dots a_{i+1}, y_{i+2} \dots y_{s-1}, z_{t-1} \dots z_0 \\ z_0 \dots a_{i+1}, z_{i+2} \dots z_{t-1}, y_{s-1} \dots y_0 \end{cases} \tag{3.25}$$

The third chain on either branch of the above factorization corresponds to the unaltered subpath. Its throughput is immutable and the denoted subpath can be instantly reused. The throughput on each subpath is evaluated independently via formula (3.6) or (3.8). As a valid subpath, this third chain can be safely removed from further consideration.

The two remaining chains form the invalid subpath. Factorization (3.25) is symmetric with respect to the subpath vertex notation and can be reformulated as indicated below:

$$\left.\begin{array}{l} y_0 \dots a_{i+1}, y_{i+2} \dots y_{s-1} \\ z_0 \dots a_{i+1}, z_{i+2} \dots z_{t-1} \end{array}\right\} = x_0 \dots a_{i+1}, x_{i+2} \dots x_{n-1} \quad \forall n = \{s|t\} \tag{3.26}$$

Through notation (3.26) the light and the eye subpath chains can be handled generically. Consequently, the above notation will be adopted throughout the rest of this analysis.

3.4.2.2 Immutable first chain throughput

Based on the path factorization, the two-chain reconstruction scenario can be refined to target an invalid path $\bar{x}_{s,t}^{\varphi_i}$ that has a valid subpath connected to a severed one. The dysfunctional subpath comprises two chains. The *first chain* runs from the first vertex of the subpath all the way to the anchor, i.e. $x_0 \dots a_{i+1}$. Conversely, the *second chain* subsumes the vertices located downstream from the anchor, i.e. $x_{i+2} \dots x_{n-1}, n = \{s|t\}$.

The first chain of the dysfunctional subpath is valid and its throughput is immutable:

$$[\eta_{i+2}(x_0 \dots x_i a_{i+1})]^{\varphi_k} = [\eta_{i+2}(x_0 \dots x_i x_{i+1})]^{\varphi_i} \tag{3.27}$$

The previous equality can be demonstrated by expanding the throughput with one step:

$$\begin{aligned} [\eta_{i+2}(x_0 \dots x_i a_{i+1})]^{\varphi_k} &= [\eta_{i+1}(x_0 \dots x_i)]^{\varphi_i} \frac{f_s(x_{i-1} \to x_i \to a_{i+1})}{P^{\perp}(x_i \to a_{i+1})} \\ &= [\eta_{i+1}(x_0 \dots x_i)]^{\varphi_i} \frac{f_s(x_{i-1} \to x_i \to x_{i+1})}{P^{\perp}(x_i \to x_{i+1})} \\ &= [\eta_{i+2}(x_0 \dots x_i x_{i+1})]^{\varphi_i} \end{aligned} \tag{3.28}$$

where scattering preservation was used in conjunction with property (3.24). For $i = 0$, f_s is substituted either by $L_e(x_0 \to a_1)$ or by $W_e(a_1 \to x_0)$ based on the subpath type. For eye subpaths, f_s must be replaced with the adjoint BSDF f_s^* to evaluate importance.

On the other hand, the second chain requires a more complex analysis due to it being severed from the first chain. To consolidate the two chains into a functional subpath and to recreate a consistent flow, a valid intra-subpath connection must be established.

3.4.2.3 Intra-subpath connectivity strategy

A viable connection could be established between the severed chains, if there existed a vertex in the second chain of the dysfunctional subpath which is visible from anchor:

$$\exists x_j \in x_{i+2} \dots x_{n-1} \quad \forall n = \{s|t\} \quad s.t. \quad \left[V\left(a_{i+1} \leftrightarrow x_j\right)\right]^{\varphi_k} = 1 \quad (3.29)$$

The above condition is a necessary, but not a sufficient, condition for the connection $a_{i+1} - x_j$ to be established. To accept a tentative connection, the scattering properties of the connecting vertices and the probabilistic nature of subpaths must be considered.

Regarding the properties of the connecting vertices a_{i+1} and x_j, the decisive condition is to ensure sufficient scattering occurs along the edge $a_{i+1} - x_j$. The evaluation of the throughput must consider the connecting edge $a_{i+1} - x_j$ both as an outgoing and as an incoming direction. Let $\Lambda_{i+1,j}$ be the contribution of the tentative connection:

$$\Lambda_{i+1,j} = f_s\left(x_i \to a_{i+1} \to x_j\right) G\left(a_{i+1} \leftrightarrow x_j\right) f_s\left(a_{i+1} \to x_j \to x_{j+1}\right) \quad (3.30)$$

The second BSDF in the above equation will be evaluated only if x_j has a successor.

To avoid inadequately small contributions without introducing bias, the tentative connection contribution $\Lambda_{i+1,j}$ is submitted to the following Russian roulette test:

$$\Lambda^*_{i+1,j} = \begin{cases} \frac{1}{q^{\Lambda}_{i+1,j}} \Lambda_{i+1,j} & \xi < q^{\Lambda}_{i+1,j} \\ 0 & otherwise \end{cases} \quad (3.31)$$

where ξ is a random number and $q^{\Lambda}_{i+1,j}$ is the probability with which $\Lambda_{i+1,j}$ is accepted:

$$q^{\Lambda}_{i+1,j} = \min\left\{1, \frac{\Lambda_{i+1,j}}{\varkappa}\right\} \quad (3.32)$$

where $\varkappa$ is chosen to reflect the throughput from the anchor onwards, had the subpath been generated through the conventional local path sampling techniques. Specifically:

$$\varkappa = \begin{cases} 10^{-(i+2)} & \exists x_{j+1} \\ 10^{-(i+1)} & otherwise \end{cases} \quad (3.33)$$

Contributions larger than $\varkappa$ are always evaluated, whereas smaller contributions are randomly discarded. Test (3.31) guarantees that the first and the second chain will be reconnected only if appropriate transport can be established along the disjoint subpath.

The second important condition for establishing the connection $a_{i+1} - x_{j}$, regards the sampling probability densities and the stochastic termination of subpaths. The sampling techniques discussed in subsection 3.1.1 generate subpaths randomly, based on locally defined probability density functions. The first technique samples a vertex on a surface using a predefined probability density function (PDF), whereas the second technique generates a subpath vertex by sampling a new direction from the local PDF. Each subpath vertex, and thus each subpath, is generated with a specific probability density.

However, the sampling techniques do not address the termination of subpaths. Without a termination criterion, subpaths would be unnecessarily long and would consume the same volume of resources irrespective of their contributions. Russian roulette solves these problematics efficiently and without introducing bias. The appropriateness of the tentative contribution $\Lambda_{i+1,j}$ is addressed by test (3.31). The stochastic feasibility of the tentative connection is established through another Russian roulette test. This second rejection test evaluates the probabilities associated with the scattering events along the connecting edge $a_{i+1} - x_{j}$ and determines whether these events can occur.

To evaluate the probabilities associated with the scattering events at a_{i+1} and x_{j}, the edge $a_{i+1} - x_{j}$ must be processed both as an outgoing and as an incoming direction. This means that the probability for scattering to occur, in and from the direction of the connecting edge, must be determined based on the probabilities $P^{\perp}(a_{i+1} \to x_{j} | x_{i})$ and $P^{\perp}(x_{j} \to x_{j+1} | a_{i+1})$. For clarity, the probabilities measured with respect to the projected solid angle indicate the other subpath vertex that conditions their definition.

Let q_{i+1} be the probability for connecting the anchor a_{i+1} to the second-chain vertex $x_{ȷ}$:

$$q_{i+1} = \min\left\{1, \frac{f_s\left(x_i \to a_{i+1} \to x_{ȷ}\right)}{P^{\perp}\left(a_{i+1} \to x_{ȷ} \middle| x_i\right)}\right\} \tag{3.34}$$

where $(1 - q_{i+1})$ is the probability with which a_{i+1} fails to be connected to vertex $x_{ȷ}$.

The connecting probability q_{i+1} suffices only if $x_{ȷ}$ does not have a successor. If $x_{ȷ}$ is followed by another vertex $x_{ȷ+1}$, then the connectivity attempt must also consider the possibility that the subpath terminate at $x_{ȷ}$ with probability $(1 - q_{ȷ})$, where $q_{ȷ}$ is the probability with which the subpath continues past vertex $x_{ȷ}$ and is defined as follows:

$$q_{ȷ} = \min\left\{1, \frac{f_s\left(a_{i+1} \to x_{ȷ} \to x_{ȷ+1}\right)}{P^{\perp}\left(x_{ȷ} \to x_{ȷ+1} \middle| a_{i+1}\right)}\right\} \tag{3.35}$$

Probabilities (3.34) and (3.35) allow the subpath to terminate both at the anchor a_{i+1} and at the vertex $x_{ȷ}$, similar to the way termination is enacted for the regular generation of subpaths. Veach (1998, p. 309) uses a similar continuation probability to control the subpath lengths. The connecting edge $a_{i+1} - x_{ȷ}$ is evaluated both as an outgoing and as incoming direction. In fact, the edge $a_{i+1} - x_{ȷ}$ causes a change in throughput along the subpath and the probabilities q_{i+1} and $q_{ȷ}$ assess the viability of this change.

Hence, the probability for the tentative connection to occur can be defined as follows:

$$q^{P}_{i+1,ȷ} = q_{i+1} q_{ȷ} \qquad \forall x_{ȷ} \neq x_{n-1} \tag{3.36}$$

The tentative connection is rejected with probability $(1 - q_{i+1} q_{ȷ})$. Condition (3.36) is a continuation test evaluated simultaneously for two subpath vertices and thus it can be interpreted as a two-dimensional form of Russian roulette (chapter 2, equation 2.64).

If the probabilities $P^{\perp}\left(a_{i+1} \to x_{ȷ} \middle| x_i\right)$ and $P^{\perp}\left(x_{ȷ} \to x_{ȷ+1} \middle| a_{i+1}\right)$ are proportional to the BSDF, then the continuation probabilities q_{i+1} and $q_{ȷ}$ are the fractions of energy scattered, rather than absorbed, for the incoming directions $x_i \to a_{i+1}$ and $a_{i+1} \to x_{ȷ}$.

Hence, if the tentative connection can be established with a probability $q_{i+1,j}^{P}$, then the probabilities used to sample the vertices x_j and x_{j+1} must be updated as shown below:

$$\begin{aligned} P^{\perp}(x_{j-1} \rightarrow x_j | x_{j-2}) \quad &= \quad q_{i+1} P_{os}^{\perp}(x_{j-1} \rightarrow x_j | x_{j-2}) \\ &and \\ P^{\perp}(x_j \rightarrow x_{j+1} | x_{j-1}) \quad &= \quad q_j P_{os}^{\perp}(x_j \rightarrow x_{j+1} | x_{j-1}) \end{aligned} \tag{3.37}$$

where $P_{os}^{\perp}$ denotes the probabilities with which vertices x_j and x_{j+1} were originally sampled. It is important to realize the nature of the intra-subpath connection. Firstly, the vertices x_j and x_{j+1} were sampled using different incident directions from the ones assumed for the tentative connection, i.e. $x_{j-2} \rightarrow x_{j-1}$ and $x_{j-1} \rightarrow x_j$ versus $x_i \rightarrow a_{i+1}$ and $a_{i+1} \rightarrow x_j$. Secondly, the meaning of the intra-subpath connection is that of generating a subpath in multiple pieces, by connecting different subpath chains. Hence, the connection does not attempt to fit a probability density function to another one, nor does it fundamentally alter the probabilities with which the vertices were sampled. The latter are simply adapted to the criteria used to establish the intra-subpath connection, just as they are adapted when subpaths are terminated via Russian roulette.

Viewing the intra-subpath connection as a method for generating subpaths, determines the approach to computing the multiple importance weights for the reconstructed paths.

The sufficient condition for setting $a_{i+1} - x_j$ is given by the connection probability:

$$q_{i+1,j} = \begin{cases} q_{i+1,j}^{\Lambda} \cdot q_{i+1,j}^{P} & \forall x_j \neq x_{n-1} \\ q_{i+1,j}^{\Lambda} \cdot q_{i+1} & x_j = x_{n-1} \end{cases} \tag{3.38}$$

The index of the connecting vertex x_j is designated as the *welding level* and it identifies the vertex where the subpath was reconnected. A pair of anchor-welding levels forms a *reconstruction level* and marks a single instance in the reconstruction of a subpath. A subpath has as many reconstruction levels as the number of instances it was invalidated. A failed intra-subpath connection assigns to that specific anchor level a welding level equal to -1. A subpath with a welding level of -1 is regenerated from its last anchor through the standard path sampling techniques (subsection 3.1.1).

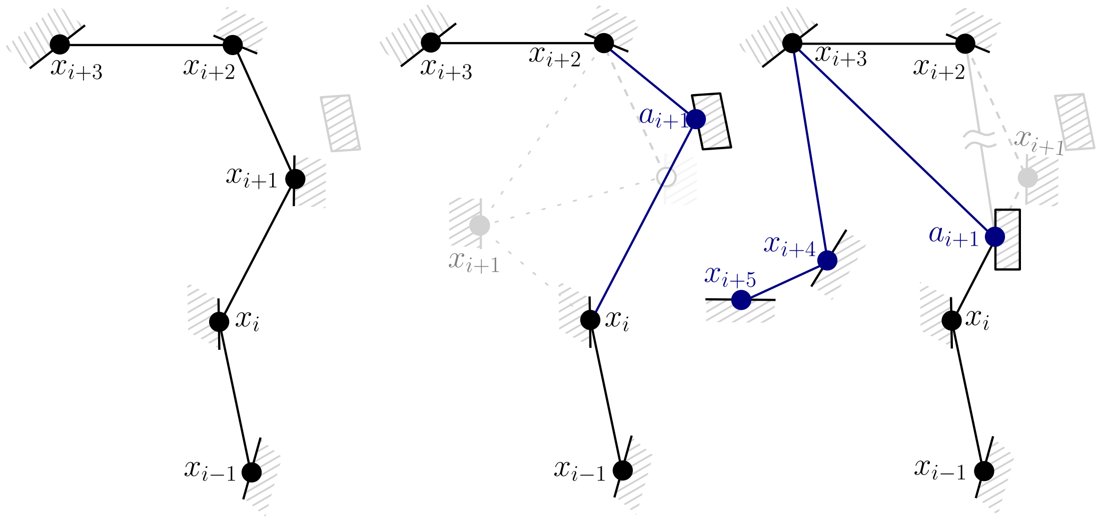


Figure 3.7: *The two-chain reconstruction scenario triggered by an in-scope (right) & an out-of-scope (centre) invalidation of a subpath (left). For the out-of-scope subpath, the anchor a_{i+1} is reconnected to the nearest vertex x_{i+2}, yielding the subpath $x_{i-1} \dots x_{i+3}$. For the in-scope subpath, the intra-subpath connection fails for vertex x_{i+2}, but succeeds for vertex x_{i+3}. The subpath is also regenerated normally from x_{i+3} to fulfil the minimum length criterion. The final reconstructed subpath is $x_{i-1} \dots x_{i+5}$.*

A successful connection allows the two severed chains to be reconnected into a single subpath of the form $x_0 \dots a_{i+1} x_j \dots x_{n-1}$. However, if the tentative connection fails then the first chain $x_0 \dots a_{i+1}$ must be regenerated from the anchor using the path sampling techniques described in subsection 3.1.1. A successfully reconnected subpath must also be regenerated from its last vertex, if its length is shorter than a minimum length. Generally, the length of a subpath is determined through the continuation probability:

$$q_i = \min\left\{1, \frac{f_s(x_{i-1} \to x_i \to x_{i+1})}{P^{\perp}(x_i \to x_{i+1})}\right\} \tag{3.39}$$

That is, a subpath is either extended past vertex x_i with probability q_i or is terminated at the same vertex with probability $(1 - q_i)$. Normally, $q_i = 1$ is imposed on the first few vertices of the subpath (Veach 1998, p. 309). The number of vertices with a fixed q_i, determines the *minimum subpath length* on which Russian roulette is not applied. Consequently, any reconnected subpath with a shorter length than the minimum one must be regenerated from its last vertex through the standard path sampling techniques.

Once reconstructed, the subpaths are used to establish bidirectional connections and the contributions of the resultant paths are evaluated as described next. Figure 3.7 illustrates the two-chain reconstruction of an in-scope and of an out-of-scope subpath.

3.4.2.4 Contribution evaluation

The current subsection investigates the contribution evaluation of those paths which have at least a subpath reconstructed via the intra-subpath connectivity strategy. Paths composed of subpaths entirely regenerated from the anchor, through the standard path sampling techniques, can be evaluated with the tools delineated in subsection 3.1.2.

The intra-subpath connection reconnects the severed chains into a functional subpath $x_0 \dots a_{i+1} x_j \dots x_{n-1}$. The first chain has an immutable throughput (equation 3.27). The throughput of the second chain can be computed based on the established connection:

$$\begin{aligned}
\left[\eta_{n-j}\left(x_j \dots x_{n-1}\right)\right]^{\varphi_k} = {} & \frac{f_s(x_i \to a_{i+1} \to x_j) \cdot G(a_{i+1} \leftrightarrow x_j)}{q^{\Lambda}_{i+1,j} \cdot P^{\perp}\left(x_{j-1} \to x_j \middle| x_{j-2}\right) \cdot G(x_{j-1} \leftrightarrow x_j)} \cdot \\
& \frac{f_s(a_{i+1} \to x_j \to x_{j+1})}{P^{\perp}\left(x_j \to x_{j+1} \middle| x_{j-1}\right)} \cdot \dots \cdot \frac{f_s(x_{n-3} \to x_{n-2} \to x_{n-1})}{P^{\perp}(x_{n-2} \to x_{n-1} | x_{n-3})} \\
= {} & \frac{f_s(x_i \to a_{i+1} \to x_j)}{q^{\Lambda}_{i+1,j} \cdot q_{i+1} \cdot P^{\perp}\left(a_{i+1} \to x_j \middle| x_i\right)} \cdot \\
& \frac{f_s(a_{i+1} \to x_j \to x_{j+1})}{P^{\perp}\left(x_j \to x_{j+1} \middle| x_{j-1}\right)} \cdot \dots \cdot \frac{f_s(x_{n-3} \to x_{n-2} \to x_{n-1})}{P^{\perp}(x_{n-2} \to x_{n-1} | x_{n-3})}
\end{aligned} \tag{3.40}$$

where the probabilities (3.37) and (3.32) were used to express the final probabilities of the connecting vertices. The geometric factors associated with vertex x_j do not cancel each other out and thus the initial probability with which x_j was sampled can be replaced with $q_{i+1} \cdot P^{\perp}\left(a_{i+1} \to x_j \middle| x_i\right)$, by virtue of the subsequent transformation:

$$\begin{aligned}
P^{\perp}\left(a_{i+1} \to x_j \middle| x_i\right) &= P^{\perp}_{os}\left(x_{j-1} \to x_j \middle| x_{j-2}\right) \frac{d\left(x_{j-1} \to x_j\right)^{\perp}}{d\left(a_{i+1} \to x_j\right)^{\perp}} \\
&= P^{\perp}_{os}\left(x_{j-1} \to x_j \middle| x_{j-2}\right) \left(\frac{d\left(x_{j-1} \to x_j\right)^{\perp}}{dA(x_j)} \cdot \frac{dA(x_j)}{d\left(a_{i+1} \to x_j\right)^{\perp}} \right) \\
&= P^{\perp}_{os}\left(x_{j-1} \to x_j \middle| x_{j-2}\right) \frac{G(x_{j-1} \leftrightarrow x_j)}{G(a_{i+1} \leftrightarrow x_j)}
\end{aligned} \tag{3.41}$$

For the vertices $x_{j+1} \dots x_{n-1}$ the projected solid angle probabilities remain unchanged.

If the reconstructed subpath has a subminimal length and is therefore regenerated from its last vertex, then the throughput on the chain $x_{n-1} \dots x_v, \forall v \geq n-1$ can be evaluated via the recursive mechanism delineated by either formula (3.6) or (3.8). Like the latter, equation (3.40) must be applied with the adjoint BSDF when evaluating eye subpaths.

The throughput on the entire, reconstructed subpath can be computed as shown below:

$$\left[\eta_{i+2+n-\jmath}(x_0 \dots a_{i+1} x_\jmath \dots x_{n-1})\right]^{\varphi_k} = \left[\eta_{i+2}(x_0 \dots a_{i+1}) \eta_{n-\jmath}(x_\jmath \dots x_{n-1})\right]^{\varphi_k} \quad (3.42)$$

The throughput of a subpath extended from its last vertex x_{n-1} can be defined similarly.

The last bit of information, required before establishing a bidirectional connection, is the probability density of the reconstructed subpath. The probability density of the first chain is easily updated via $[p_{i+2}(x_0 \dots a_{i+1})]^{\varphi_k} = P_A(x_0) \dots P_A(a_{i+1})$. The probability density of the second chain can be defined using the projected solid angle probabilities:

$$\begin{aligned} \left[p_{n-\jmath}(x_\jmath \dots x_{n-1})\right]^{\varphi_k} &= q^{\wedge}_{i+1,\jmath} \cdot q_{i+1} \cdot P^{\perp}(a_{i+1} \to x_\jmath | x_i) \cdot G(a_{i+1} \leftrightarrow x_\jmath) \quad \cdot \\ &\quad\; q_\jmath \cdot P^{\perp}_{os}(x_\jmath \to x_{\jmath+1} | x_{\jmath-1}) \cdot G(x_\jmath \leftrightarrow x_{\jmath+1}) \quad \cdot \\ &\quad\; \vdots \\ &\quad\; P^{\perp}(x_{n-2} \to x_{n-1} | x_{n-3}) \cdot G(x_{n-2} \leftrightarrow x_{n-1}) \\ &= P_A(x_\jmath) P_A(x_{\jmath+1}) \dots P_A(x_{n-1}) \end{aligned} \quad (3.43)$$

Consequently, the probability density of the entire subpath can be computed as follows:

$$\left[p_{i+2+n-\jmath}(x_0 \dots a_{i+1} x_\jmath \dots x_{n-1})\right]^{\varphi_k} = P_A(x_0) \dots P_A(a_{i+1}) P_A(x_\jmath) \dots P_A(x_{n-1}) \quad (3.44)$$

Equations (3.42) and (3.44) render the subpath amenable to bidirectional connections.

The light transport path, which is obtained by connecting the reconstructed subpath to another homologue, can be evaluated with the tools discussed in subsection 3.1.2. Because the intra-subpath connection is essentially a strategy for generating a subpath in multiple pieces, it does not affect the evaluation of the sample contribution. That is, a vertex is always evaluated either from the direction of the light source or from the direction of the camera. The group of sampling techniques is not altered by the path generation method. Hence, the weights can be computed normally via equation (3.16).

The sufficient condition (3.38) ensures that only subpaths with appropriate throughput are generated by the intra-subpath connectivity strategy. Subpaths which fail to comply with the sufficient condition are sampled over the path space through the conventional techniques. As a subpath generation method, the intra-subpath connectivity strategy does not restrict the sampling of the path space. Subpaths are generated in multiple pieces, only if the scattering (3.32) and the probabilistic (3.36) conditions are fulfilled.

Hence, the contribution of a reconstructed path can be evaluated via equation (3.4). The correctness of the generated result depends on the complete identification of the invalid paths. That is, all the invalid paths must be identified and their subpaths must be appropriately processed so that they can form viable bidirectional connections. The next two chapters further analyse the identification and reconstruction of invalid paths.

3.4.3 Terminus anchor reconstruction

The terminus anchor scenario reconstructs the in-scope and the out-of-scope subpaths that are disrupted along their last edge. That is, the reconstruction target is any subpath with an a_{s-1} or an a_{t-1} anchor. Such an anchor is generated when the invalidation scenario (3.23) is verified on the edge $x_i - x_{i+1}, i = \{s|t\} - 2$. As the terminus of the invalid subpath, the anchor no longer divides the latter into two dysfunctional chains:

$$x_0 \dots x_{n-2} a_{n-1} \qquad \forall n = \{s|t\} \tag{3.45}$$

The above chain defines the entire structure of the invalid subpath. Designated as a *single-chain subpath,* such an invalid subpath is a special case of the two-chain subpath. Thus, the terminus anchor scenario is a simplification of the two-chain reconstruction.

Just like in the previous scenario, the invalidation affects only one of the subpaths. Hence, the invalid subpath can be reconstructed independently of the valid subpath, which can be immediately reused to construct and evaluate a new light transport path.

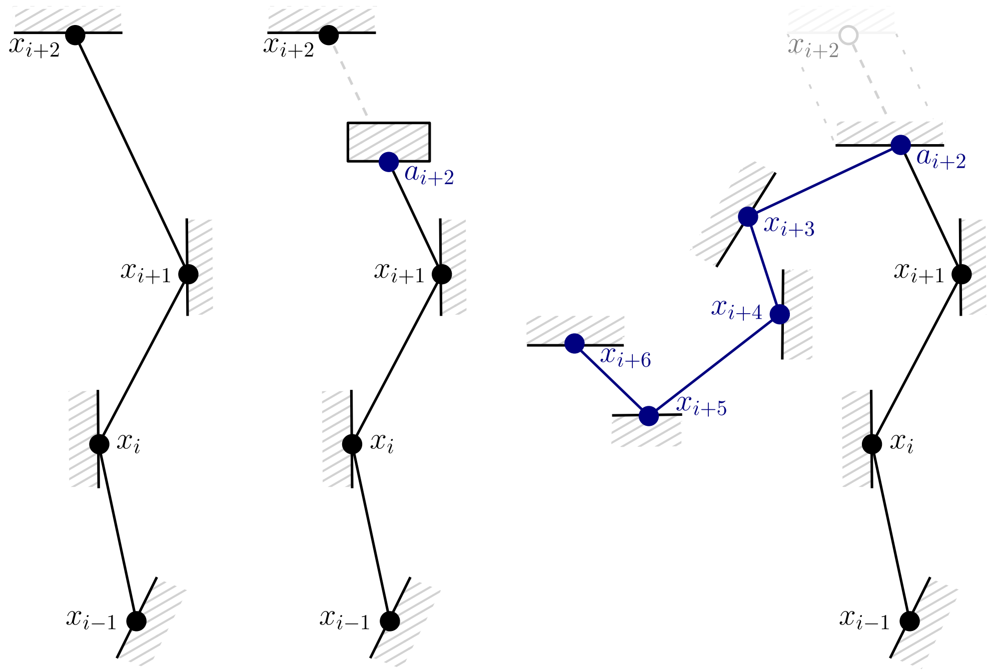


***Figure 3.8**: The terminus anchor reconstruction triggered by an in-scope (centre) and an out-of-scope (right) invalidation of a subpath (left). The in-scope invalidation shows a_{i+2} as the intersection between an occluder and the direction $\vec{\omega}_{i+1} = \widehat{x_{i+2} - x_{i+1}}$. The single-chain subpath $x_{i-1} \dots a_{i+2}$ is terminated at a_{i+2} with probability $(1 - q_{i+2})$. In the out-of-scope invalidation vertex x_{i+2} moves backwards along the direction $\vec{\omega}_{i+1}$, generating the anchor a_{i+2}. The subpath is extended conventionally from a_{i+2} to x_{i+6}.*

The terminus anchor preserves the structural unity of the invalid subpath and just like the first chain (equation 3.27) the single-chain subpath has an immutable throughput:

$$[\eta_n(x_0 \dots x_{n-2}a_{n-1})]^{\varphi_k} = [\eta_n(x_0 \dots x_{n-2}x_{n-1})]^{\varphi_i}$$

The probability density of a single-chain subpath can be computed easily, as follows:

$$[p_n(x_0 \dots x_{n-2}a_{n-1})]^{\varphi_k} = P_A(x_0) \dots P_A(x_{n-2})P_A(a_{n-1})$$

Formulae (3.24) and (3.7) fully define the surface area probability associated with a_{n-1}.

However, a single-chain subpath $x_0 \dots x_{n-2}a_{n-1}$ must be given the opportunity to be extended from its anchor. As the subpath is extended from a_{n-1}, through the standard path sampling techniques discussed in subsection 3.1.1, the minimum subpath length and the random termination criteria are fulfilled through the continuation probability (3.39). The throughput on the new portion of the subpath can be evaluated with the tools described in subsection 3.1.2. Once reconstructed, a single-chain subpath can be used to construct and evaluate a new light transport path. Figure 3.8 illustrates the terminus anchor reconstruction applied on an in-scope and on an out-of-scope subpath.

```
for all paths
    for $i = 0, s + t - 1$
        if $(i = 0) \wedge (x_0\ moved)$                                   //primary anchor scenario
            sample new vertex $x_0$
            generate new direction $\vec{\omega}_0 = \widehat{x_1 - x_0}$
            compute $\tau(x_0, \vec{\omega}_0) = a_1$                        //equation (3.23)
        if $(i < s - 1) \vee (i < t - 1) \vee (x_0\ moved)$                //two-chain scenario
            if $(i < s - 1) \vee (i < t - 1)$
                if $(x_i\ moved)$                                          //out-of-scope invalidation
                    $\vec{\omega}_i = \widehat{x_i - x_{i-1}}$
                else                                                       //in-scope invalidation
                    $\vec{\omega}_i = \widehat{x_{i+1} - x_i}$
                compute $\tau(x_{i-1/i}, \vec{\omega}_i) = a_{i/i+1}$
            determine visible vertex $x_{\jmath}$                           //equation (3.29)
            compute $q^{\wedge}_{i+1,\jmath}$ , $q_{i+1}$ and $q_{\jmath}$   //equations (3.32), (3.34) and (3.35)
            if $\left(q^{\wedge}_{i+1,\jmath} > \xi_1 \wedge q_{i+1}q_{\jmath} > \xi_2 \cdot \xi_3\right) \vee \left(q^{\wedge}_{i+1,\jmath} > \xi_1 \wedge q_{i+1} > \xi_2\right)$
                $x^*_{0,n-1}$ = reconnect subpath chains $x_0 \dots a_{i+1}x_{\jmath} \dots x_{n-1}, \forall n = \{s|t\}$
            if $(i + 2 + n - \jmath < minimum\ length) \vee (reconnection\ failed)$
                $x^*_{0,n-1}$ = regenerate conventionally from last vertex $x_{n-1}/a_{i+1}$
            evaluate throughput $\eta_n\left(x^*_{0,n-1}\right)$               //equations (3.27), (3.40) and (3.42)
            compute $p_n(x^*_{0,n-1})$                                     //equations (3.7), (3.24), (3.43) and (3.44)
            reuse $x^*_{0,n-1}$
        if $(i = s - 1 \vee i = t - 1) \wedge (x_i\ moved)$               //terminus anchor scenario
            $\vec{\omega}_{i-1} = \widehat{x_i - x_{i-1}}$
            compute $\tau(x_{i-1}, \vec{\omega}_{i-1}) = a_i$
            $x^*_{0,n-1}$ = regenerate conventionally from $a_i$ where $n - 1 \geq i$
            evaluate throughput $\eta_n\left(x^*_{0,n-1}\right)$ and compute $p_n(x^*_{0,n-1})$
            reuse $x^*_{0,n-1}$
        if $(V(x_{s-1} \leftrightarrow x_{t-1})\ has\ changed)$            //collateral identification
            randomly connect $(x_0 \dots x_{s-1})^L$ to eye subpaths
            $\bar{x}$ = connect $(x_0 \dots x_{t-1})^E$ to a random light subpath
            evaluate the contribution of $\bar{x}$                          //equation (3.5)
        reuse the valid subpath to which $x^*_{0,n-1}$ was originally connected
    end for
end for
```

***Figure 3.9**: A high-level description of the propounded path manipulation algorithm.*

3.5 High-level algorithm description

Sections 3.2 - 3.4 analysed the major steps of the path manipulation algorithm. This section uses the outlined steps to provide a high-level description of the proposed algorithm. The algorithmic precis relies on the information synthesized in Figure 3.9.

For a collection of paths, the path manipulation algorithm first checks (line 3) whether there are subpaths with initial vertices displaced by the movement of the camera or of a light source. The primary anchor reconstruction (subsection 3.4.1) starts by sampling a new vertex (line 4) and a new direction (line 5) for each out-of-scope subpath. The primary anchor is given by the closest ray-primitive intersection (line 6). Using the intra-subpath connectivity strategy (subsections 3.4.2.3 - 3.4.2.4), the path manipulation algorithm completes the reconstruction (lines 14 - 22) of a first-vertex invalid subpath.

Subsequent invalidations of the subpaths are processed with the two-chain scenario (subsection 3.4.2). As discussed in section 3.3, an out-of-scope subpath is identified by tracing a ray from the previous vertex towards the old position of the dynamic vertex (line 10). An in-scope subpath is identified by tracing a ray in the direction of the assessed edge (line 12). If the algorithm identifies a visible second-chain vertex (equation 3.29) for the computed anchor (line 13), then it computes the probabilities required for the establishment of the intra-subpath connection. The contribution of the tentative connection is subjected to the rejection test (3.31), whereas its stochastic viability is evaluated based on the probability (3.36). If both Russian roulette tests are successful (line 16), then the subpath chains are reconnected (line 17). The condition displayed on line 16, represents the implementation of the sufficient condition (3.38). Like the sufficient condition, the implementation discriminates between multi-vertex and single-vertex second chains. It also evinces the two-dimensional aspect of the rejection test (3.36). Whenever the subpath length is subminimal or the intra-subpath connection fails, the subpath is conventionally regenerated (subsection 3.1.1) from its last vertex (line 19). By evaluating its throughput (line 20) and its probability density (line 21), the algorithm entirely reconstructs and renders reusable the invalid subpath.

The terminus anchor scenario (subsection 3.4.3) reconstructs only the out-of-scope subpaths with a dynamic last vertex. All the other subpaths are reconstructed with the two-chain scenario (lines 7 - 22). The terminus anchor reconstruction (lines 23 - 28) first computes the anchor along the ray traced from the penultimate vertex to the old position of the dynamic last vertex. Then, it conventionally regenerates the subpath from the anchor (line 26) and assesses its throughput and probability density (line 27).

The path manipulation algorithm finishes by identifying and evaluating the collateral paths (lines 29 - 32). Conceptually, the collateral paths are identified by checking the visibility of the connecting edge (section 3.3). Specific approaches to identifying the collateral paths are discussed in chapter 4 (subsection 4.3.5). The collateral subpaths are reused to construct and evaluate complete light transport paths (subsection 3.1.2).

3.6 Algorithm analysis

The performance extremes of the path manipulation algorithm can be analysed using worst-case and best-case scenarios. One category of worst-case scenarios occurs when all the eye subpaths must be both reconstructed and re-evaluated, as is the case of the camera transformation. In such a case, the path manipulation algorithm must resample all the primary rays and recompute all the primary anchors (Figure 3.9, lines 4 - 6). The number of subpaths reconnected via the intra-subpath connection depends on the scattering properties of the connecting vertices and on the probability for scattering to occur in and from the direction of the connecting edge (Figure 3.9, line 16). Hence, the intra-subpath connectivity varies across scenes and it impacts both the retracing of rays and the conventional regeneration of subpaths. The latter (Figure 3.9, line 19) is obviated with each successful, non-subminimal, intra-subpath connection. When an intra-subpath connection fails, the second-chain vertices to which an anchor failed to be connected are no longer considered and thus the retracing of their associated rays is circumvented. However, each unconnected subpath is conventionally regenerated from its anchor. A high number of unconnected primary anchors affects performance, since the algorithm must entirely regenerate those subpaths. Performance decreases also with the exhaustive re-evaluation of the eye subpaths (Figure 3.9, lines 20 - 22).

Concrete performance and reconstruction analysis, for two camera transformation examples, is carried in chapter 5 (section 5.3 and subsections 5.5.4 - 5.5.5). Compared to the path manipulation algorithm, the spatio-temporal architecture proposed by Havran et al. (2003) attains superior reuse for camera transformation, by reprojecting the primary samples of full eye subpaths across a range of previous and subsequent frames. Yet, the weighting heuristic adopted by the latter approach biases the solution. Moreover, light and eye subpaths are conventionally regenerated from the points where they intersected dynamic objects. The path manipulation algorithm reconstructs these subpaths as well, with the advantage that the secondary chains of the invalid subpaths can be effectively reused. Besides, it can operate without predefined animation paths.

Méndez-Feliu et al. (2006) enhance the spatio-temporal architecture with an estimator that unbiasedly combines reprojected eye subpaths and supports non-diffuse materials. However, this approach not only requires predefined animation paths, but also restricts dynamism to camera transformation. Consequently, it surpasses the path manipulation algorithm only under such conditions and fails to propound a path reuse strategy for other types of transformation, such as those involving scene objects or light sources.

Another worst-case scenario for the path manipulation algorithm occurs when the only light source of the scene is transformed. Such a case entails the reconstruction of all the light subpaths, the reconstruction of the in-scope and out-of-scope eye subpaths and the re-evaluation of all the eye subpaths. The alteration of the sole light source disrupts all the light subpaths and the path manipulation algorithm must reconstruct them via the primary anchor and two-chain scenarios (Figure 3.9, lines 3 - 22). The light source transformation also disrupts some of the generated eye subpaths, which must be reconstructed with the two-chain scenario (Figure 3.9, lines 7 - 22). Though the remaining eye subpaths preserve their structures, they are identified as collateral since their contributions change due to the reconstruction of all the light subpaths. The high-level description of the path manipulation algorithm handles such eye subpaths implicitly (Figure 3.9, line 33). Hence, all the eye subpaths must be re-evaluated. An example of a single light source transformation is analysed in chapter 5 (section 5.2).

Relative to the path manipulation algorithm, the strategy devised by Sbert et al. (2004a) always reuses whole light subpaths, by reconnecting the dynamic light source to the primary samples of the light subpaths generated in the anterior and posterior frames. However, this approach necessitates predefined animation paths and is restricted to light source transformations. Therefore, it does not provide equivalent strategies to those used by the path manipulation algorithm for the reconstruction and reuse of invalid subpaths ascribable to camera or other object transformations. The algorithm proposed by Sbert et al. (2004b) simply limits the reuse of light subpaths to preceding frames and thus adapts the previous strategy to interactive applications. Hence, in terms of reuse these two advancements do not depart considerably from each other. Neither does the method proposed by Sbert and Castro (2004), which adopts the same strategy for reusing light subpaths. Given the assumptions of predefined animation paths, static camera and static objects, this approach also reuses gathering subpaths of length 1, to accumulate the radiosity from all other frames at the current samples. The path manipulation algorithm can reuse subpaths beyond these working assumptions.

The best-case scenarios for the path manipulation algorithm are those transforming other objects than the camera and involving multiple light sources. A dynamic object invalidates less subpaths than a dynamic camera and does not cause the exhaustive reconstruction and re-evaluation of subpaths. Similarly, the transformation of a light source in a multi-light scene invalidates a limited number of subpaths. In such a case, the directly invalidated subpaths are those affiliated with the transformed light source and those disrupted by its geometry. The indirectly invalidated subpaths are the ones whose contributions change due to their connections with the altered light source and its affiliated subpaths. Hence, the path manipulation algorithm must reconstruct and/or re-evaluate only the disrupted and collateral subpaths. Performance in such scenarios is drawn from both the algorithmic strategies and the lower number of processed subpaths. As discussed for the first worst-case scenario, the intra-subpath connectivity impacts the retracing of rays (Figure 3.9, line 13) and the conventional regeneration of subpaths (Figure 3.9, lines 18 - 19). Chapter 5 analyses the factors that influence the intra-subpath connectivity (subsection 5.1.3), the implementation strategies that are performance sources (section 5.6) and concrete examples (sections 5.1, 5.4 - 5.5).

The algorithm propounded by Lai (2010) exhibits similar capabilities to the path manipulation algorithm, in that it can reuse paths invalidated anywhere along their lengths. A path is reused across a range of frames via a temporal perturbation, which rigidly transforms the first two vertices of the path and then extends the latter either through specular bounces or by rigidly transforming the diffuse vertices. Light source, object and camera transformations can all be supported. Practically though, changes in illumination are processed only by cutting the animation into smaller sequences, which conforms with the presupposition of low variation in the lighting condition. Another requirement of this algorithm concerns the predefined animation paths. Still, its reuse per path can be higher than for the path manipulation algorithm, because the reconstruction of a path is conditioned only by the visibility between the transformed vertices. The path manipulation algorithm considers not only the visibility along the intra-subpath connection, but also its scattering and stochastic properties. Yet, the path manipulation algorithm has the advantage of being applicable to interactive systems.

The cost of rendering a frame without path reuse, through bidirectional path tracing is:

$$\underbrace{N_\tau(\varsigma_{\mathcal{r}} + \varsigma_\tau + \varsigma_\eta)(N_{EP} + N_{LP})}_{\substack{subpath \\ generation}} + \underbrace{N_\tau^2 \varsigma_{eval} N_{EP}}_{\substack{path \\ contribution}} \tag{3.46}$$

where N_τ is the average number of vertices per subpath, $\varsigma_{\mathcal{r}}$ is the cost of sampling a ray, ς_τ is the cost of computing the first scene intersection by tracing a ray, ς_η is the cost of evaluating the throughput of a vertex, ς_{eval} is the cost of evaluating the contribution of a path constructed by connecting an eye and a light subpath vertex, N_{EP} is the number of eye subpaths and N_{LP} is the number of light subpaths. The cost of the first subpath vertex was assumed to be equivalent to that of the other vertices, although it is generated by sampling a predefined surface distribution. For a successful bidirectional connection, the evaluation cost ς_{eval} also includes the cost of assessing the visibility between the eye and light vertices. Otherwise, it reduces to the latter. Since additional samples are extracted by connecting each eye vertex to all the vertices of a light subpath, there are a total of $N_\tau \times N_\tau$ contribution evaluations per eye subpath.

The cost of rendering a frame with the path manipulation algorithm in the worst-case, when the camera transformation yields a majority of unconnected primary anchors, is:

$$\underbrace{N_\tau\big(\varsigma_r + \varsigma_\tau + \varsigma_\eta\big)N_{EP}}_{\substack{eye\ subpath\\ regeneration}} \quad + \quad \underbrace{\varsigma_{recon}N_{ILP}}_{\substack{light\ subpath\\ reconstruction}} \quad + \quad \underbrace{N_\tau^2\varsigma_{eval}N_{EP}}_{\substack{path\\ contribution}} \tag{3.47}$$

which is comparable to the cost of bidirectional path tracing, with the difference that the reconstruction of N_{ILP} invalid light subpaths is less expensive than the complete regeneration of all the light subpaths. When the camera is not modelled as part of the scene, it does not invalidate the light subpaths and the cost for their reconstruction reduces completely. The reconstruction cost per eye/light subpath can be estimated via:

$$\varsigma_{recon} = \underbrace{N_a\big(N_\jmath\varsigma_\tau + \big(N_\tau - N_\jmath\big)\varsigma_\eta\big)}_{\substack{intra-subpath\\ connection}} + \underbrace{(N_\tau - N_v - 1)\big(\varsigma_r + \varsigma_\tau + \varsigma_\eta\big)}_{\substack{subpath\\ regeneration}} \tag{3.48}$$

where $N_\jmath$ is the average number of vertices up to $x_\jmath$ (equation 3.29) for which a ray is traced and N_v is the average number of vertices preceding the vertex from which the subpath is conventionally regenerated. The vertices $x_{\ell+1} \dots a_{i+1}x_{i+2} \dots x_{\jmath-1}$ only require that rays be traced, without the need to sample those rays or evaluate the throughput of the vertices. The rays are traced between vertices, so they need not be sampled. The vertices $x_\ell \dots a_{i+1}$ represent a fist chain with immutable throughput (equation 3.27). Vertex x_ℓ refers to x_0 for the first invalidation of the subpath and to the first vertex succeeding the penultimate anchor for any subsequent invalidations. The vertices $x_{i+2} \dots x_{\jmath-1}$ represent the second-chain vertices to which the anchor failed to be connected and thus they only require that visibility rays be traced between them and the anchor a_{i+1}. The connecting vertex $x_\jmath$ requires both a visibility ray originating at the anchor and the evaluation of its throughput. Hence, the cost of reconstructing the subpath up to $x_\jmath$ is $N_\jmath\varsigma_\tau$. Since the anchor a_{i+1} is connected to $x_\jmath$, the vertices $x_{\jmath+1} \dots x_{N_\tau-1}$ only require that their throughput be updated, without any retracing involved. Hence, the cost of reconstructing the subpath from $x_\jmath$ is $\big(N_\tau - N_\jmath\big)\varsigma_\eta$. On average a subpath is assumed to be invalidated N_a times, which corresponds to the average number of anchors per subpath. As such, a multiply invalidated subpath has an intra-subpath connection cost of $N_a\big(N_\jmath\varsigma_\tau + \big(N_\tau - N_\jmath\big)\varsigma_\eta\big)$. When a subpath has an unconnected anchor or a subminimal length, it will be conventionally regenerated from the vertex with index $v = \{i + 1|N_\tau - 1\}$. Hence, the conventional regeneration cost is $(N_\tau - N_v - 1)\big(\varsigma_r + \varsigma_\tau + \varsigma_\eta\big)$. Adding the last two costs yields the cost (3.48).

Using the reconstruction cost per subpath, the rendering cost of the path manipulation algorithm in the case when the only light source of the scene is transformed, becomes:

$$\underbrace{\varsigma_{recon}(N_{LP} + N_{IEP})}_{\substack{subpath\\ reconstruction}} + \underbrace{N_\tau^2 \varsigma_{eval} N_{EP}}_{\substack{path\\ contribution}} \tag{3.49}$$

where the limited number of reconstructed eye subpaths N_{IEP} contrasts the total number of light subpaths which undergo reconstruction. As explained for the second worst-case scenario, all the eye subpaths are re-evaluated. In all the other scenarios, the rendering cost of the path manipulation algorithm can be deduced via equation (3.49) by substituting N_{LP} with N_{ILP} and N_{EP} with N_{IEP}. The reconstruction and/or re-evaluation of a subset of disrupted and collateral subpaths augments performance.

However, the reconstruction cost per subpath (3.48) is a loose upper bound since not every subpath needs to be conventionally regenerated. Besides, a failed intra-subpath connection entails a purely tracing cost $(N_\tau - \ell - 1)\varsigma_\tau$ instead of the tracing-evaluation cost $N_j \varsigma_\tau + (N_\tau - N_j)\varsigma_\eta$. A more accurate reconstruction cost per subpath is given by:

$$\underbrace{\sum_{1}^{N_a^\#} (j - \ell)\varsigma_\tau + \delta_{i+1,j}\left(\varsigma_{q_{i+1,j}^\Lambda} + \varsigma_\eta\right) + (N_\tau - j - 1)\left(\delta_{i+1,j}\varsigma_\eta + (1 - \delta_{i+1,j})\varsigma_\tau\right)}_{intra-subpath\ connection}$$

$$+$$

$$\underbrace{(1 - \delta_{i+1,j} + \delta_{i+1,j}\delta_{v,\tau})(N_\tau - N_v^\# - 1)(\varsigma_r + \varsigma_\tau + \varsigma_\eta)}_{subpath\ regeneration} \tag{3.50}$$

where $N_a^\#$ is the actual number of invalidations per subpath, $\varsigma_{q_{i+1,j}^\Lambda}$ is the cost of computing the probability $q_{i+1,j}^\Lambda$ (equation 3.32) and $\delta_{i+1,j}$ is defined as follows:

$$\delta_{i+1,j} = \begin{cases} 1 & a_{i+1} - x_j \ is\ established\ with\ q_{i+1,j} \\ 0 & otherwise \end{cases} \tag{3.51}$$

where $q_{i+1,j}$ is defined by equation (3.38). The value of $\delta_{i+1,j}$ determines which terms in equation (3.50) are evaluated. The vertices $x_{\ell+1} \ldots x_j$ only require that rays be traced. If the intra-subpath connection $a_{i+1} - x_j$ is established, then the throughput at vertex x_j is evaluated with cost $\varsigma_{q_{i+1,j}^\Lambda} + \varsigma_\eta$ and the throughput of the vertices $x_{j+1} \ldots x_{N_\tau - 1}$ is evaluated with cost $(N_\tau - j - 1)\varsigma_\eta$. When the intra-subpath connection fails, the latter cost changes from a throughput evaluation cost to a tracing cost $(N_\tau - j - 1)\varsigma_\tau$.

A subpath with an unconnected anchor ($\delta_{i+1,j} = 0$) or a subminimal length ($\delta_{v,\tau} = 1$) will be regenerated from its last vertex. The subminimal length case is expressed via:

$$\delta_{v,\tau} = \begin{cases} 1 & N_v^{\#} < N_\tau \\ 0 & otherwise \end{cases} \tag{3.52}$$

where $N_v^{\#}$ is the actual number of vertices preceding the vertex from which the subpath is conventionally regenerated. Condition $(1 - \delta_{i+1,j} + \delta_{i+1,j}\delta_{v,\tau})$ ensures that the subpath is regenerated only in one of the two cases mentioned above. Still, both an unconnected subpath and a successfully reconnected, subminimal subpath are regenerated with a cost equal to $(N_\tau - N_v^{\#} - 1)(\varsigma_r + \varsigma_\tau + \varsigma_\eta)$. Hence, the overall reconstruction cost per subpath is the cost of a single regeneration added to the sum of all reconnection costs.

The images rendered with the path manipulation algorithm may suffer from bias if the collateral paths are inadequately reconstructed. The methods delineated in chapter 4 (subsection 4.3.5), may not suffice in some cases and certain collateral paths will not be adapted to the scene configuration. The example analysed in chapter 5 (section 5.4) is such a case. However, the strong intuition is that the reconstruction of collateral paths can be stabilised through further investigation. Tables 2.1 and 2.2 contextualise the path manipulation algorithm using various criteria, including rendering artefacts.

3.7 Conclusions

This chapter discussed the theoretical foundation of the path manipulation algorithm. Defined as an apparatus of sampling and reuse strategies, path manipulation addresses the restriction of static geometry imposed on Monte Carlo light transport simulations.

Despite generating a variety of illumination effects for generic scene configurations, bidirectional path tracing fails to accommodate scene dynamism. The slightest change of a scene causes the full recomputation of the illumination solution. The disposal of paths, immediately after the evaluation of their contributions, limits the path lifespan to a generation-evaluation cycle and imposes a static use of the sampling techniques.

The path manipulation algorithm supplants the static manipulation of paths with a generation-evaluation-reuse cycle and ports bidirectional path tracing to the temporal domain. By reconstructing and reusing paths, the novel algorithm supports dynamism.

The concepts of immutable contribution and path validity (section 3.2) defined the norms that identify the subpaths which can be immediately reused in the rendering of the altered scene. Path invalidation and anchor computation (section 3.3) identified the subpaths that breach the validity criteria and must be reconstructed for ensuing frames.

Based on the location of the anchor on an invalid subpath, reconstruction (section 3.4) branched into three scenarios. The primary and terminus anchor reconstructions treated the antipodal cases when the anchor either follows the first vertex (subsection 3.4.1) or replaces the last vertex (subsection 3.4.3) of an invalid subpath. The two-chain reconstruction (subsection 3.4.2) treated subpaths split in two chains around the anchor.

The foremost contribution is the elaboration of the intra-subpath connectivity strategy (subsection 3.4.2.3), which reconnects two dysfunctional chains of the same type into a functional subpath. The novel strategy uses the scattering and stochastic properties of the connecting vertices to set an intra-subpath connection. Tentative contributions with insufficient throughput are discarded without introducing bias (test 3.31). Also, the stochastic viability of the tentative connection is determined via a two-dimensional rejection test (3.36). Unlike conventional path generation methods, which use Russian roulette only to control the subpath lengths, the current approach uses such tests also throughout the connectivity process. A successful intra-subpath connection generates a subpath in more than one piece by reusing existent path information. Consequently, the two-chain reconstruction is a path generation approach that maximizes path reuse.

The outcome is the extension of bidirectional path tracing towards reusing paths in the temporal domain. As demonstrated in section 3.4, the path manipulation algorithm reconstructs the light and eye subpaths generically, without regard to whether a light source, the camera or another object was transformed. Moreover, predefined animation paths are not required. Chapter 4 delineates the implementation of the algorithm and chapter 5 examines cases in which the algorithm can operate without animation paths.

Chapter 4

Light transport framework implementation

The practical goal of the current work is to implement a Monte Carlo light transport framework that uses path manipulation strategies to simulate a wide variety of illumination effects in conditions of dynamic geometry. Defined as an apparatus of sampling and reuse strategies, path manipulation obviates the regeneration of the entire collection of light transport paths, by reconstructing only those paths that are invalidated by the changes of the scene. The result is a Monte Carlo algorithm that reduces the heavy computational load generally associated with this class of methods.

This chapter discusses the implementation details of the path manipulation algorithm. The proposed algorithm extends bidirectional path tracing towards reusing paths in the temporal domain. The path manipulation strategies, investigated in the previous chapter, reconstruct subpaths independently of their type and of the scene dynamism.

The first section of this chapter presents the algorithms implemented in the light transport framework. The next section examines the architecture of the framework and identifies the building blocks that define the pipeline of both the standard and the path manipulation algorithm. The schematic flow of bidirectional path tracing concludes this second section. The third section recasts the standard Monte Carlo pipeline to accommodate the path manipulation strategies. A new building block is used, together with the existing ones, to fully describe the reconstruction process. The essential difference between the standard and the reconstruction pipeline refers to whether a collection of light transport paths is discarded or preserved between frames. This discrimination governs the variations and the similarities between the two pipelines. The chapter concludes with the schematic flow of the path manipulation algorithm.

4.1 Supported Monte Carlo algorithms

The light transport framework supports three Monte Carlo ray tracing algorithms. These algorithms are variants of published light transport methods and they reflect the modular implementation of the light transport framework. Each algorithm is obtained by varying the functionality of the building blocks that define the Monte Carlo pipeline.

The first algorithm generates a collection of light subpaths and uses it to establish bidirectional connections with eye subpaths of length 1. Every prefix of a light subpath is connected to a primary sample and the contribution of the resultant path is accrued at the apposite pixel. This approach is similar to instant radiosity (Keller 1997), except that pseudorandom numbers replace the low-discrepancy sequences used to sample the virtual point lights and path contributions are evaluated using Monte Carlo integration.

The second algorithm is a path tracer with next event estimation (Kajiya 1986; Dutré and Willems 1995), which samples a group of eye subpaths, connects each eye subpath vertex to each light source and evaluates the overall radiance scattered towards the eye.

Bidirectional path tracing (Veach 1998, p. 297) is the third Monte Carlo algorithm implemented in the light transport framework. As a generalization of the other two algorithms, it generates paths by connecting each vertex of an eye subpath to all the vertices of a randomly selected light subpath. The path manipulation strategies process the light and eye subpaths generically and irrespective of the scene dynamism type. Hence, the extension of bidirectional path tracing to the temporal domain emphasizes the generality of the path manipulation algorithm. Path reuse methods usually require animation paths, restrict dynamism or fail to reconstruct both light and eye subpaths. Previous work is scrutinised in chapter 2 (subsection 2.6.3) and chapter 3 (section 3.6).

The path manipulation strategies are not restricted to a bidirectional approach but can be used to extend most path reliant algorithms towards reusing samples in the temporal domain. That is, path manipulation can be used with gathering and shooting methods.

4.2 The standard Monte Carlo pipeline

The light transport framework is structured on building blocks, so as to easily define the pipelines of the standard and reconstruction algorithms. The rationale behind the building blocks is to easily change the behaviour at one stage of the pipeline or to replace an entire building block with another. The standard Monte Carlo pipeline consists of three phases: path generation, contribution evaluation and image synthesis.

Path generation starts by tracing packets of primary rays and by computing the first scene intersections. From the primary samples, subpaths are extended progressively by sampling the local probability densities and by marching the generated rays in parallel. Subsequently, the complete subpaths enter the contribution evaluation phase. The first task of the estimator block is to establish connections using the information provided by the visibility tester. The contributions of the light transport paths are then evaluated according to the specific Monte Carlo algorithm. The image synthesis phase uses the radiance stored at the roots of the eye subpaths to generate the current frame. Figure 4.1 illustrates the building blocks used in the standard Monte Carlo pipeline. Note that every iteration of the rendering process fully regenerates the path collection.

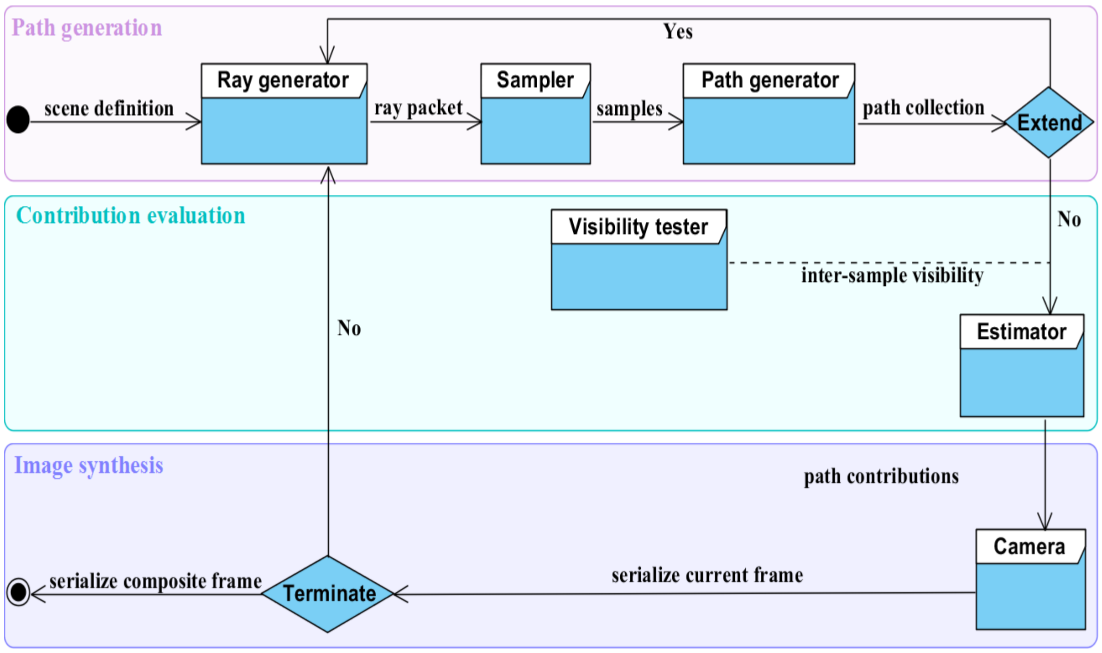


***Figure 4.1**: The three major phases that constitute the standard Monte Carlo pipeline.*

The modular design of the light transport framework has two important advantages. The first advantage is flexibility in defining the standard Monte Carlo algorithms, whereas the second advantage is augmentation in deriving the path manipulation algorithm. The standard Monte Carlo algorithms are implemented by simply varying the functionality of certain building blocks, while the novel algorithm is implemented by merely replacing some of the building blocks utilized in the path generation phase.

The framework also has a hybrid architecture, which combines techniques from both offline and progressive Monte Carlo light transport simulation, with the purpose of improving accuracy as well as performance. The used offline techniques refer mainly to efficient sample generation and variance reduction. The progressive techniques are targeted at expediting the execution. Together, offline and progressive optimization techniques determine the structure of the building blocks comprised in the framework.

The blocks of the contribution evaluation and image synthesis phases execute the same tasks in the path manipulation algorithm, while the blocks of the path generation phase serve the path reconstructor block (section 4.3) in replacing the generation of paths with their reconstruction. Thus, the building blocks described next also serve to define the pipeline of the path manipulation algorithm. The reconstruction pipeline is derived from the standard Monte Carlo pipeline by simply replacing the path generation phase.

Thus, in terms of design, modularity is the substance of the light transport framework. From a code perspective, the light transport framework entails several contributions. As the first one accomplished within Optis, the implementation of bidirectional path tracing is a contribution. Another contribution is the first in-house implementation of the recursive multiple importance sampling schema (van Antwerpen 2011b). The adaptation of the path generation method proposed by Novák et al. (2010) completes the list of first in-house implementations. The version developed in the light transport framework terminates subpaths stochastically, rather than tracing them up to a user predefined length and then extending them up to a maximum length derived from the average scene reflectance. The essence of this work is the path manipulation algorithm and as such the path reconstructor is the crux in the implementation of the framework.

4.2.1 Ray generation

The ray generator starts the execution of the standard pipeline by generating packets of primary rays. Depending on the algorithm, the primary rays are generated from the camera (path tracing), from the light sources (light tracing) or from both directions (bidirectional path tracing). Throughout the path generation phase, the ray generator samples the rays used to progressively extend the unterminated subpaths. If multiple subpaths are used per light source or per pixel, then the generator also spawns the primary rays for the new subpaths that replace the terminated ones. For the evaluation of the path contributions, the generator supplies the visibility tester with shadow rays.

Generally, the number of rays traced per frame amounts to orders of millions. To avoid prohibitive execution costs, without sacrificing the flexibility of the modular approach, the GPU is used as a ray tracing processor. The ray generator is capable of spawning large batches of rays, either directly on the GPU or on the CPU. The primary rays that originate at the camera and cross the image plane, are generated directly on the device using a dedicated CUDA kernel (Sanders and Kandrot 2012). The rays that start at the light sources, extend subpaths, restart new subpaths or assess visibility are generated and packed on the host. Yet, all rays are traced on the GPU. To ensure a transparent communication between the device and the host, memory allocations, data formatting and copy operations are abstracted in a designated memory management class (buffer).

The generated rays must form packets large enough to ensure an appropriate workload for the GPU. A high rate of device occupancy results in a good utilization of the GPU resources and in an equipoise to the costs associated with the transfer of data over the PCIe bus. Most ray generation routines produce packets of rays. When a calling function requests a single ray from the generator, it will preserve the ray packet approach by accumulating multiple such isolated rays before transferring them to the ray tracing engine. Tracing packets of rays means that the path generation process concurrently extends multiple subpaths, with one segment at a time, by marching the sampled rays in parallel. In other words, the path generator does not extend one subpath at a time until it terminates, rather it extends batches of subpaths in parallel.

When a subpath terminates and the requested number of samples is not reached, that subpath will be automatically replaced with a new one, to retain a compact marching of paths for as long as possible. This path generation process is similar to the one proposed by Novák et al. (2010), except that no limitation is imposed on the lengths of subpaths.

4.2.2 Point sample computation

The sampler receives the packets of rays from the ray generator and determines the closest intersections between the received rays and the scene geometry. The computed intersections are the next point samples of those subpaths to which the rays correspond.

The ray generator catalyses the tracing of rays on the device by generating the primary rays with CUDA kernels. However, the sampler is the core of the hybrid design. One task performed by the sampler is the computation of the intersections between the geometry and the ray packets received from the ray generator. The sampler creates an OptiX Prime query, associates the ray buffer with the query, specifies the format of the intersection buffer and instructs the Prime API (NVIDIA 2017) to execute the query. Prime builds the ray acceleration structure, performs the ray traversal, computes the primitive intersections and populates the intersection buffer with the tracing results. The ray traversal and intersections are executed on the device. The sampler brings the intersection data on the host, computes the samples and appends them to the subpaths.

4.2.3 Path generation

The path generator represents the main processing loop of the path generation phase and it coordinates the generation, extension, termination and regeneration of subpaths.

First, the renderer instructs the ray generator to produce a batch of primary rays and to initialize the corresponding subpaths by creating their roots. A *root* is the very first vertex of a subpath, sampled directly either on a light source or on the camera lens.

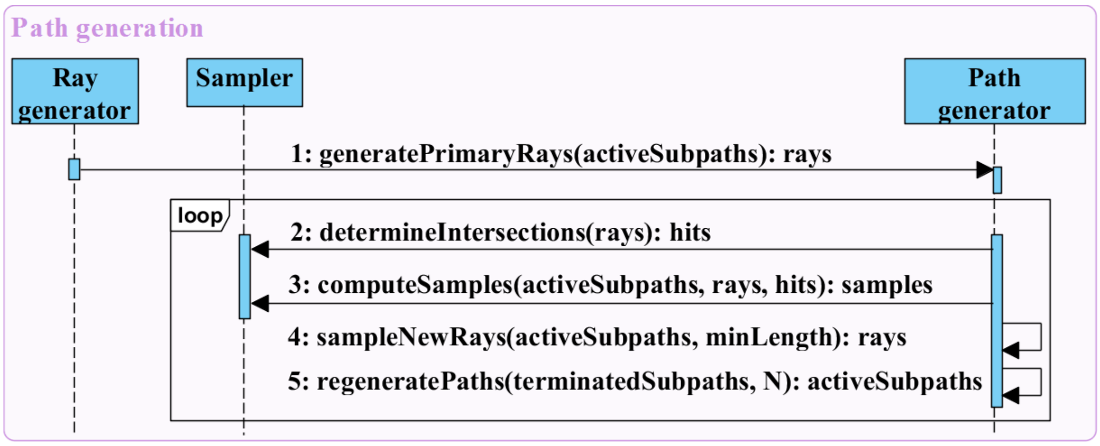

Path generation
Ray generator
Sampler
Path generator
1: generatePrimaryRays(activeSubpaths): rays
loop
2: determineIntersections(rays): hits
3: computeSamples(activeSubpaths, rays, hits): samples
4: sampleNewRays(activeSubpaths, minLength): rays
5: regeneratePaths(terminatedSubpaths, N): activeSubpaths

***Figure 4.2**: The generation of paths depicted as a processing loop of the path generator.*

Then, the renderer launches the path generation loop by invoking the path generator with the buffer of initialized subpaths and the batch of primary rays. The path generator sends the subpaths and the primary rays to the sampler and waits for the latter to trace the rays and to compute the primary samples. Using the BSDFs of the primary vertices, the path generator samples the rays that will determine the next vertices of the subpaths.

The subpaths whose rays did not intersect the geometry or which failed to satisfy the continuation criteria are terminated. For each terminated subpath, the path generator will restart another one, if the requested number of samples is not exceeded. These operations, except for the initialization performed by the ray generator, are repeated until no active subpath is left in the queue of the processing loop. An *active subpath* is an unterminated subpath. Figure 4.2 details the events handled by the path generator.

The path generator maximizes the device performance by marching in parallel as many rays as feasible. To maintain a compact grid of rays for as long as possible, the path generator alternates the extension of the active subpaths with the regeneration of the terminated ones. After sampling the new rays (Figure 4.2, step 4), the path generator separates the active subpaths from the terminated ones and executes the regeneration routine (Figure 4.2, step 5). The active subpaths and their next-bounce rays are kept in the same buffer, whereas the terminated subpaths are stored in the buffer that will be dispatched to the contribution evaluation routine. The locations of the terminated subpaths in the active buffer are filled with new subpaths that have identical affiliations to the terminated ones. Every new subpath is assigned both a root and a primary ray.

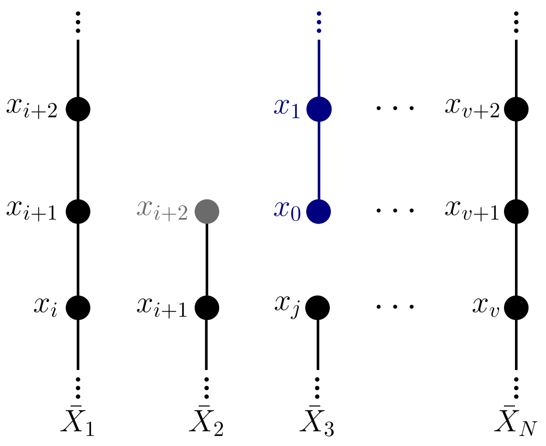


***Figure 4.3**: The generation of N subpaths. Subpath $\bar{X}_2$ randomly terminates at vertex x_{i+2} and no other subpath is restarted in its place, as the required number of samples for its affiliation is reached. The grid of subpaths will be compacted to $N-1$ subpaths. Subpath $\bar{X}_3$ terminates at x_j and a new subpath is restarted in its place. Subpaths $\bar{X}_1$ and $\bar{X}_N$ are extended from vertex x_{i+1}, respectively x_{v+1}, by sampling the local BSDF.*

The primary ray is generated by invoking the ray generator with the corresponding light source or pixel. The *affiliation* is a pair that stores the identifier of a light source or the coordinates of a pixel and associates a subpath with a specific light source/pixel. The affiliation is used to keep track of the actual number of subpaths generated per light source or per pixel. If the number of subpaths generated for a given affiliation is equal to the specified threshold, then the regeneration routine will supress the creation of a new subpath and will compress the active subpath buffer. By reducing the grid of active subpaths, the path generation loop does not benefit from the same device performance throughout its execution. As the number of regenerated subpaths declines, the number of idle threads increases and performance drops. However, the goal is to extend bidirectional path tracing towards reusing paths in the temporal domain and not to develop techniques targeted at obtaining an optimal performance from the GPU.

The path generator retains a compact grid of rays by also bundling individual rays. The extension routine (Figure 4.2, step 4) packs all the next-bounce rays sampled by the ray generator. The regeneration routine (Figure 4.2, step 5) interleaves the buffer of extension rays with the primary rays generated for the newly restarted subpaths. Each ray in the current bundle is mapped to its corresponding active subpath. Once packed, the extension and primary rays are sent to the sampler to be traced on the GPU. Figure 4.3 illustrates the extension of subpaths with the regeneration of new subpaths.

Though implementing progressive strategies, the path generator is not a pure GPU approach to sampling paths. Unlike the method proposed by Novák et al. (2010), the path generator does not condition the lengths of the subpaths, but terminates the latter stochastically. The extension routine is responsible for sampling the next-bounce rays and for terminating subpaths. To sample the next-bounce rays, the extension routine invokes the ray generator with the last vertex of each active subpath. The ray generator samples a new propagation ray based on the local probability density function of the last subpath vertex. Each propagation ray has a specific probability and the extension routine uses this probability in a Russian roulette test, to determine the length of a subpath. The continuation probability $q_i = \min\left\{1, \frac{f_s(x_{i-1} \to x_i \to x_{i+1})}{P^{\perp}(x_i \to x_{i+1})}\right\}$ is used to decide whether a subpath should be extended past or be terminated at vertex x_i. Generally, $q_i = 1$ is imposed on the first few vertices of each subpath, to avoid any extra variance caused by short subpaths (Veach 1998, p. 309). The number of vertices with fixed q_i, determines the *minimum subpath length* on which Russian roulette in not applied. The extension routine takes a parameter that specifies the minimum subpath length and uses it to suppress Russian roulette. A subpath terminates in three other circumstances.

One such circumstance refers to a propagation ray failing to intersect the geometry. In this case, the ray exits the scene and does not produce the next subpath vertex. Total absorption also causes a subpath to terminate, as does the random intersection of a light source to an eye subpath. The latter case regards a path sampled with zero vertices on the light subpath via naïve path tracing (i.e. $s = 0$). When all the active subpaths terminate, the path generation loop exits and the contribution evaluation phase begins.

4.2.4 Visibility computation

The visibility tester computes visibility either between point samples or between light sources and point samples. For the path tracer, it computes the visibility between each vertex of an eye subpath and the light sources present in the scene. For the light tracing algorithm and for bidirectional path tracing it computes both types of visibility, i.e. light sources – point samples and inter-sample. Like the path generator, the visibility tester performs its computations by simultaneously processing multiple paths. Given a buffer of paths, the tester assesses visibility breadthwise across all paths, rather than depth-wise along a single path at a time. For bidirectional path tracing, this approach facilitates the efficient extraction of additional samples from multiple paths at once. Veach (1998, p. 300) suggested that additional samples can be efficiently generated from the same path, by joining each prefix of the light subpath to each suffix of the eye subpath. The tester computes a one-to-one vertex visibility for many paths at once.

Bidirectional path tracing entails the most generic approach to computing visibility. Given a buffer of eye subpaths, the renderer selects as many random light subpaths as the number of eye subpaths. Then, the tester determines the visibility between the primary samples of the eye subpaths and the first vertices of the light subpaths. All the eye subpaths whose assigned light subpaths still have vertices are selected and the tester assesses the visibility between their primary samples and the secondary vertices of the light subpaths. This process continues until all the primary samples have been connected to all the vertices of their assigned light subpaths. Then, the renderer selects the next level of eye subpath vertices. A *vertex level* refers to all the vertices in a subpath buffer, which have the same index. For example, the i^{th} vertex level refers to all the vertices in a subpath buffer which have the i^{th} index in their subpaths. To every selected vertex, the render assigns a new random light subpath and the tester assesses the visibility between each eye vertex and all the vertices of its allotted light subpath. That is, the current level of eye vertices is progressively connected to all the levels of the assigned light subpaths. The visibility computation ends when all the vertices of the eye subpaths have been processed. Figure 4.4 illustrates the visibility computation between each vertex of an eye subpath and all the vertices of the allotted light subpath.

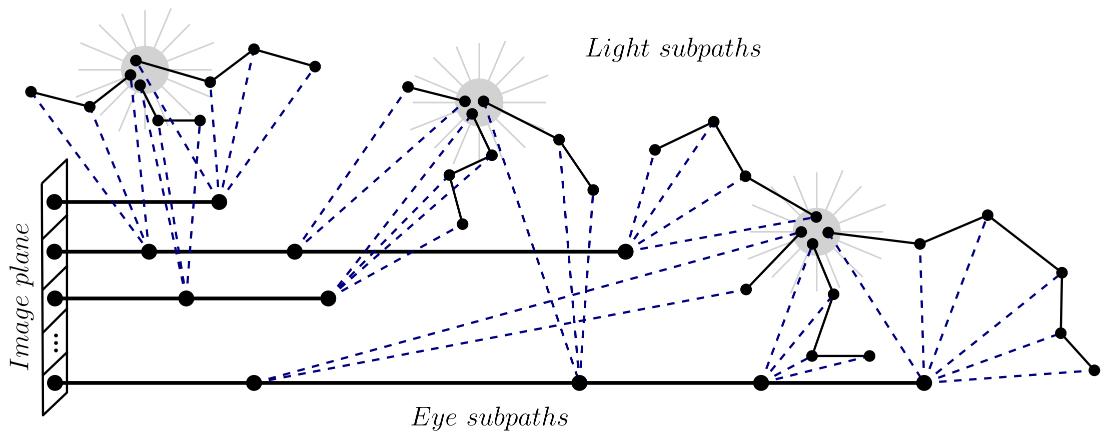


***Figure 4.4**: The bidirectional visibility (navy dashed lines) is determined by tracing shadow rays between each vertex of the eye subpath and every vertex of the randomly assigned light subpath. Unobstructed shadow rays represent feasible path connections.*

The light tracing algorithm specializes the bidirectional visibility computations, in that only the primary samples need to be connected to randomly assigned light subpaths. As discussed in section 4.1, this approach generates eye subpaths of length $k = 1$ and progressively connects the primary samples to all the vertices of the allotted light subpaths. The path tracing algorithm progressively selects the vertex levels of a grid of eye subpaths and invokes the tester to compute the visibility between each vertex in the selected level and all the light sources present in the scene. All the visibility information is computed by tracing shadow rays between the eye and the light vertices.

For light tracing and for bidirectional path tracing, the visibility tester invokes the ray generator with the buffers of light and eye subpaths and with a vertex level for each buffer. For path tracing, the visibility tester invokes the ray generator with the set of light sources and with the buffer of eye subpaths and its associated vertex level. The ray generator computes the shadows rays. The visibility tester dispatches the shadow ray buffer to the sampler, which traces the rays and determines their intersections with the geometry. The resultant intersection buffer contains the visibility information that the estimator uses to establish bidirectional connections. Hence, the visibility tester is as a lightweight wrapper class that abstracts the details of the visibility computations. All paths, except for the unidirectional paths, require visibility computations when establishing their connections. Figure 4.5 illustrates the visibility computation process.

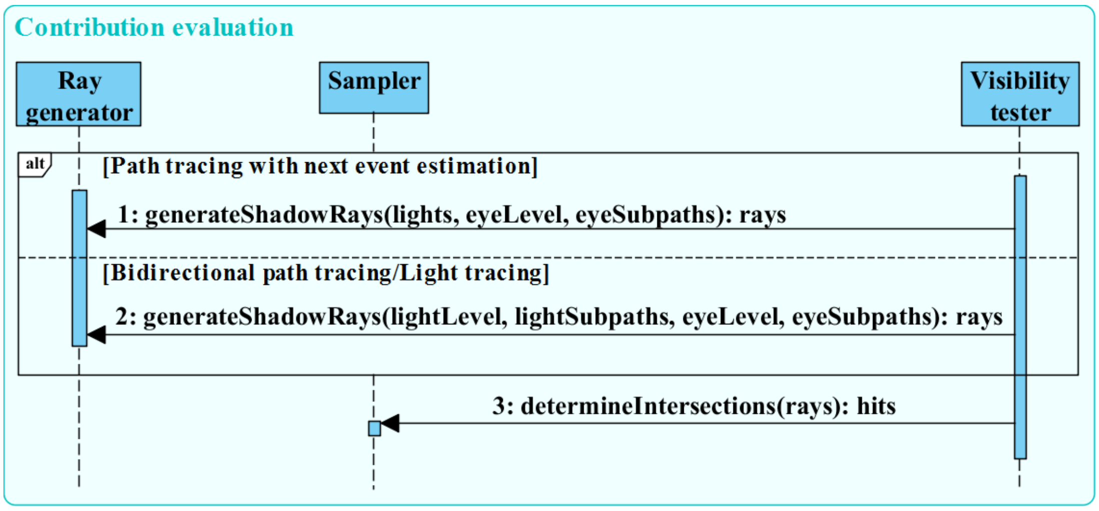

Contribution evaluation
Ray generator
Sampler
Visibility tester
alt
[Path tracing with next event estimation]
1: generateShadowRays(lights, eyeLevel, eyeSubpaths): rays
[Bidirectional path tracing/Light tracing]
2: generateShadowRays(lightLevel, lightSubpaths, eyeLevel, eyeSubpaths): rays
3: determineIntersections(rays): hits

Figure 4.5: *The different operations executed by the tester when determining visibility.*

4.2.5 Contribution evaluation

The estimator is the engine of the contribution evaluation phase. It efficiently draws multiple samples from the generated paths (Veach 1998, p. 300), evaluates the path contributions and computes the power heuristic weights (Veach and Guibas 1995).

The most basic approach to sampling multiple paths $\bar{x}_{s,t}$ is to generate a separate light and eye subpath for each path. A path $\bar{x}_{s,t}$ is randomly sampled with s vertices on the light subpath and t vertices on the eye subpath and has an associated probability density $p_{s,t}$. The subpath connection is established between the vertices y_{s-1} and z_{t-1}. Additional paths can be generated from the same pair of light and eye subpaths, by connecting each prefix of the light subpath to each suffix of the eye subpath. Like this, several paths can be extracted at once from a whole group of sampling techniques $p_{s,t}$.

This path generation approach has several advantages. Firstly, path information is reused among the paths generated from the same subpath pair. Secondly, the additional paths are sampled with zero costs. Lastly, the use of multiple samples reduces variance.

Like the method proposed by Novák et al. (2010), the estimator connects each vertex of an eye subpath to a randomly assigned light subpath. Multiple paths are extracted at once, by connecting each eye vertex to all the vertices of the assigned light subpath.

There are several advantages to connecting each eye vertex to a new light subpath. The paths extracted from the same pair of subpaths are correlated. By increasing the number of independently generated paths, this correlation decreases. The random permutation of the light subpaths among the eye subpath vertices further reduces the correlation. Novák et al. (2010) observe that the deterministic noise, caused by connecting the same light subpath to all the vertices of an eye subpath, decreases by permuting the light subpaths among the eye vertices (*intra-subpath permutation*) and implicitly across the different eye subpaths (*inter-subpath permutation*). The rationale is that the random assignment of the low and the high contribution subpaths distributes the radiance of the complete light transport paths more evenly across the image plane.

For bidirectional path tracing, the renderer selects as many random light subpaths as the number of eye subpaths (Figure 4.6, step 2). Then, the visibility tester computes the intersection buffer for the primary samples of the eye subpaths and the first vertices of the light subpaths (Figure 4.6, step 3). The estimator uses the intersection buffer to establish bidirectional connections and to evaluate the contributions of the resultant paths (Figure 4.6, step 4). Subsection 3.1.2 of the previous chapter minutely discusses the evaluation of the path contribution. Next, only the primary samples whose light subpaths have unprocessed vertices (Figure 4.6, step 5) are connected to the next level of light vertices. The visibility and the evaluation steps are repeated until all the vertex levels of the light subpaths are exhausted. Then, the renderer selects the next level of eye subpath vertices (Figure 4.6, step 1) and repeats the entire visibility-evaluation process (Figure 4.6, steps 2 - 5). The evaluation loop exits when all the eye vertices have been processed. Figure 4.6 illustrates the entire contribution evaluation process.

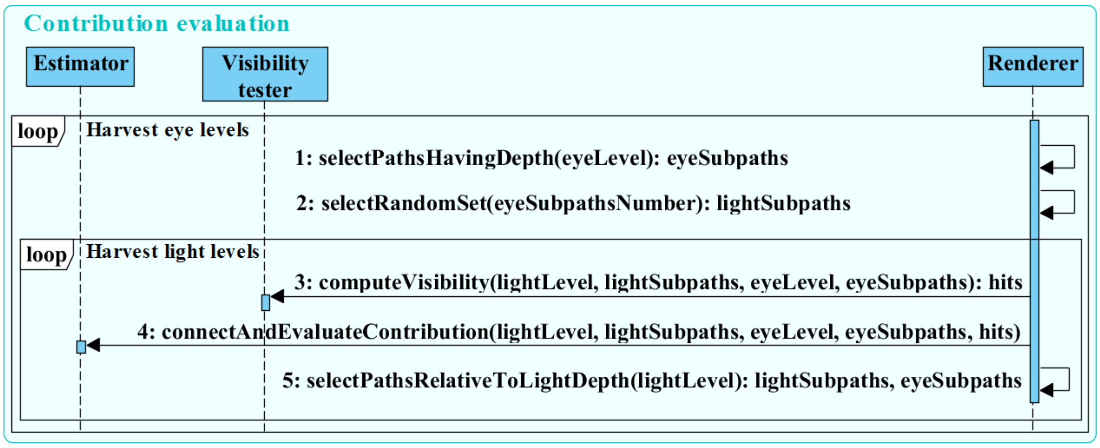

Contribution evaluation
Estimator
Visibility tester
Renderer
loop
Harvest eye levels
1: selectPathsHavingDepth(eyeLevel): eyeSubpaths
2: selectRandomSet(eyeSubpathsNumber): lightSubpaths
loop
Harvest light levels
3: computeVisibility(lightLevel, lightSubpaths, eyeLevel, eyeSubpaths): hits
4: connectAndEvaluateContribution(lightLevel, lightSubpaths, eyeLevel, eyeSubpaths, hits)
5: selectPathsRelativeToLightDepth(lightLevel): lightSubpaths, eyeSubpaths

Figure 4.6: *The processing loop which forms and evaluates full light transport paths.*

The evaluation routine (Figure 4.6, step 4) computes the contribution $C_{s,t} = w_{s,t}\eta_s c_{s,t}\eta_t$ (chapter 3, equation 3.5). The term η_s is the throughput on the light subpath and η_t is the throughput on the eye subpath. The throughput is computed and stored at each vertex during the subpath generation. If two subpaths can be successfully connected, then the evaluation routine computes $c_{s,t}$ and the weight $w_{s,t}$. Two methods are used to compute the power heuristic weight with an exponent $\beta = 2$. One method aims to reduce the memory footprint, whereas the other aims to reduce the execution time.

The first method computes the weights using the approach proposed by Veach (1998). Besides the throughput, each subpath vertex stores the geometric factor G, the forward probability and the reverse probability. The *forward probability* is the projected solid angle probability with which a vertex was sampled, i.e. $P^{\perp}(x_{i-1} \to x_i), \forall i \geq 1$. The *reverse probability* is the projected solid angle probability with which a vertex could have been sampled from the opposite direction, i.e. $P^{\perp}(x_{i+1} \to x_i), \forall i < \{s|t\} - 1$. Pharr and Humphreys (2004, p. 663) discuss concrete methods for computing the vertex probabilities. All three terms are used to compute the probability density ratios discussed in chapter 3 (equation 3.14). The probability density ratios regard the other techniques that could have sampled the given path. By parsing both subpaths in the reverse direction, all the probability density ratios associated with the $k + 2$ sampling techniques are computed and summed to the final weight (chapter 3, equation 3.16). This weight computation method consumes slightly less memory than the ensuing one.

The second method, minimizes the work required to compute the multiple importance sampling (MIS) weights, by storing partial weights at each subpath vertex. Based on the recursive MIS computation proposed by van Antwerpen (2011b), this method computes the partial, power heuristic weights up to the antepenultimate vertex. The partial weights are still computed using the probability density ratios. Yet, the partial weight of each vertex also depends on the weight of the prior vertex ($w_{i-1} = 0, i = 0$). Like the throughput, the MIS weights are computed and stored at each subpath vertex:

$$\begin{aligned} w_0 &= \left(\frac{P^{\perp}(x_1 \to x_0)G(x_0 \leftrightarrow x_1)}{P_A(x_0)}\right)^{\beta} && i = 0 \\ w_i &= \left(\frac{P^{\perp}(x_{i+1} \to x_i)G(x_i \leftrightarrow x_{i+1})}{P^{\perp}(x_{i-1} \to x_i)G(x_{i-1} \leftrightarrow x_i)}\right)^{\beta} (1 + w_{i-1}) && \forall i \geq 1 \end{aligned} \tag{4.1}$$

The partial weights of the last two vertices of a subpath, are computed based on the established connection. The connecting edge is used both as an outgoing and as an incoming direction in the computation of the vertex probabilities. For a path $\bar{x}_{s,t}$ connected between the light vertex y_{s-1} and the eye vertex z_{t-1}, the following probabilities are computed using the connecting edge $y_{s-1}z_{t-1}$: $P^{\perp}(z_{t-1} \to y_{s-1}|z_{t-2})$, $P^{\perp}(y_{s-1} \to y_{s-2}|z_{t-1})$, $P^{\perp}(y_{s-1} \to z_{t-1}|y_{s-2})$ and $P^{\perp}(z_{t-1} \to z_{t-2}|y_{s-1})$. When the path has a single vertex on one of the subpaths, the fourth or the second probability need not be computed, whereas the first or the third probability is computed as if the direction $z_{t-1} \to y_{s-1}$ or $y_{s-1} \to z_{t-1}$ emanated from the camera, respectively from the light source. If the path is sampled with zero vertices on one of the subpaths, then the last subpath vertex is assumed to have been sampled on the lens/light source and the first/third probability is computed just like for the single vertex subpath. With these probabilities, the partial weights of the last subpath vertices w_{s-1} and w_{t-1} can finally be computed, yielding the total weight $w_{s,t} = 1/(1 + w_{s-1} + w_{t-1})$. The evaluation routine finishes by adding the weighted contributions to the roots of the eye subpaths.

There are several specializations of the contribution evaluation process. Firstly, the eye subpath vertices that randomly intersect the light sources are processed separately from the rest of the paths, as they do not require the connectivity logic. The throughput and the MIS weight can be readily obtained from the given eye subpath, with the result that the evaluation of the path contribution reduces to a product between the MIS weight, the throughput of the eye subpath and the emitted radiance of the intersected light source. These paths are sampled with naïve path tracing and they are designated as zero light vertices ($S0$) subpaths. The vertices of an $S0$ subpath, which precede the vertex that intersected the light source, are processed normally with the evaluation routine. Bidirectional path tracing connects each eye vertex to a random light subpath.

The light tracing algorithm represents another special evaluation case, in that eye subpaths are generated with length $k = 1$ and thus only the primary samples need to be connected to random light subpaths. Still, the evaluation loop remains unmodified.

The path tracing algorithm specializes more acutely the evaluation process. It only connects the eye subpath vertices to the light sources and in doing so it eliminates the inner loop of Figure 4.6, which harvests the light vertex levels. Hence, the evaluation process reduces to executing steps 1, 3 and 4. Step 4 regards the evaluation routine, which also reduces to connecting the eye subpath vertices to the light sources and to estimating the next events. Given an eye subpath vertex z_i, the next event is computed as $f_j(y_0 z_i \dots z_0) = L_e(y_0 \rightarrow z_i) G(y_0 \leftrightarrow z_i) f_s(y_0 \rightarrow z_i \rightarrow z_{i-1}) \eta_{i+1}(z_i)$, where y_0 is a vertex on a light source and η_{i+1} is the throughput evaluated at z_i (chapter 3, equation 3.8).

4.2.6 Image synthesis

The camera is used in the image synthesis phase to construct and serialize the frames. Its task is to parse the buffer of eye subpaths and to assign the radiance of each subpath root to the affiliated pixel. The current frame is serialized and composited with the prior frames. If the renderer must execute additional iterations, the pipeline will resume with the path generation phase, otherwise the composite frame will also be serialized.

With the image synthesis phase, the standard Monte Carlo pipeline comes to an end.

4.2.7 Bidirectional path tracing

Figure 4.7 illustrates the schematic flow of the implemented bidirectional path tracer.

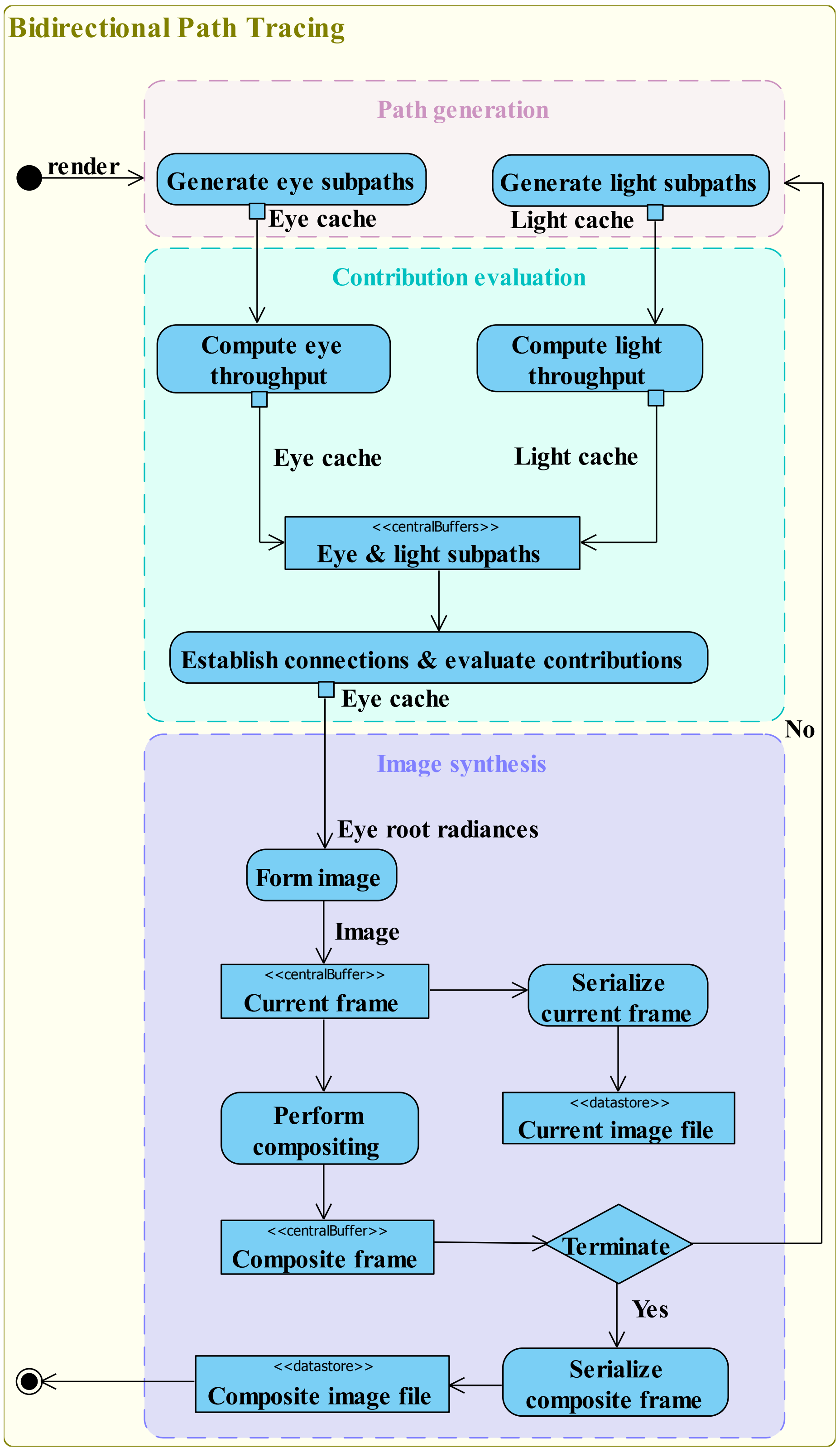


Figure 4.7: *Bidirectional path tracing defined via the standard pipeline functionality.*

The bidirectional path tracing algorithm begins by instructing the ray generator to sample the primary and the light source rays. Then, it invokes the path generator (Figure 4.2) with the buffer of primary rays, respectively with the buffer of light source rays. Once the path generator sampled the requested number of eye and light subpaths, the bidirectional path tracer invokes the estimator to compute the G factors, the reverse probabilities and the partial MIS weights for all the subpaths in each buffer. Next, the estimator connects the eye and the light subpaths and evaluates the contributions of the resultant paths (Figure 4.6). Finally, the radiance values stored at the roots of the eye subpaths are used to generate the current frame. The latter is serialized and composited with the previous frames. If the bidirectional path tracer must execute other iterations, the entire process will recommence, otherwise the composite frame will be serialized.

The flow presented in Figure 4.7 can be easily modified to define both the path tracer and the light tracing algorithm, as both are specializations of the bidirectional approach. Subsections 4.2.4 and 4.2.5 discussed the specializations entailed by these algorithms.

4.3 The reconstruction pipeline

The practical goal of the current work is to extend bidirectional path tracing towards reusing paths in the temporal domain, with the purpose of supporting scene dynamism.

Due to its modular design, the standard Monte Carlo pipeline (Figure 4.1) can fit the path manipulation strategies with a minimal change to its overall structure. The main difference between the standard and the reconstruction pipeline stands in the use that is made of paths between frames. The standard pipeline regenerates the light transport paths with each frame, whereas the reconstruction pipeline preserves the paths and adapts them to the new scene configuration. In fact, the path manipulation algorithm substitutes only the path generation phase of the standard pipeline, leaving the other two phases unaltered. The remainder of this chapter examines the building block that implements the path manipulation strategies and defines the reconstruction pipeline.

4.3.1 Invalidation and anchor computation

The reuse of paths in conditions of dynamic geometry, entails the reconstruction of the subpaths that were invalidated by the transformations of the geometry, in such a way that they become consistent with the new scene configuration. The first step in the reconstruction process is to identify the invalid subpaths. A path unaltered by the geometric transformations has an immutable contribution and can be instantly reused.

The building block that identifies and reconstructs the invalid subpaths is *the path reconstructor*. Chapter 3 (section 3.3) defined two types of invalid subpaths, namely the in-scope and the out-of-scope subpaths. An in-scope subpath is a subpath with at least one edge disrupted by the interposition of an object. An out-of-scope subpath is a subpath disrupted by the egress of one of its vertices from within its scope. The path reconstructor identifies the in-scope subpaths by searching for occluded edges. The out-of-scope subpaths are determined by searching for dynamic subpath vertices. One identification routine is used per type of invalid subpath. Also, the path reconstructor processes the light subpath buffer independently of the analogous eye subpath buffer.

Like the other building blocks, the path reconstructor determines the invalid subpaths by concurrently testing multiple subpaths. The invalidation and the reconstruction of subpaths progress one subpath edge at a time across multiple subpaths, rather than exhaustively on a single subpath at a time. That is, the breadthwise processing of paths is maintained. The reconstruction routine starts by selecting all the subpaths with a given vertex level. Then, it invokes the two identification routines with the resultant subpath buffer. Figure 4.8 depicts the identification of both types of invalid subpaths.

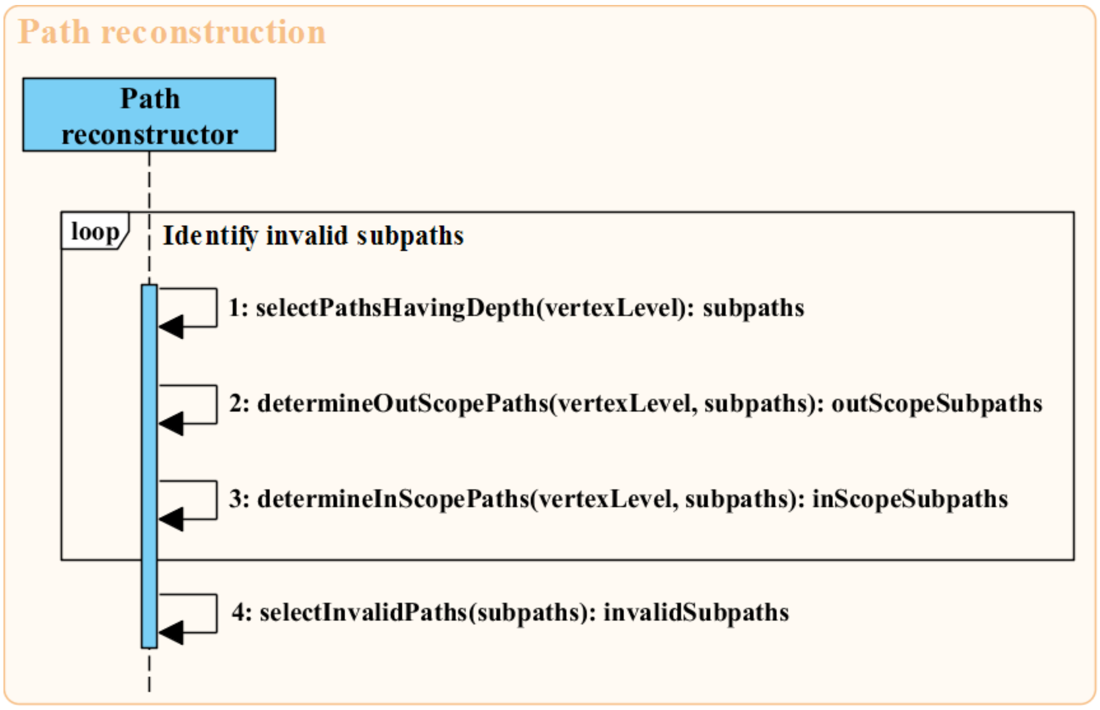

Path reconstruction
Path reconstructor
loop
Identify invalid subpaths
1: selectPathsHavingDepth(vertexLevel): subpaths
2: determineOutScopePaths(vertexLevel, subpaths): outScopeSubpaths
3: determineInScopePaths(vertexLevel, subpaths): inScopeSubpaths
4: selectInvalidPaths(subpaths): invalidSubpaths

Figure 4.8: *The identification process of the in-scope and of the out-of-scope subpaths.*

The in-scope identification routine (Figure 4.8, step 3) determines the invalid subpaths by computing the visibility along the edges that connect the current vertices to their successors. The current vertices are simply the vertices included in the current vertex level. Before computing visibility, the initial subpath buffer is parsed once more to identify all the subpaths which have at least one vertex level past the current one. The new subpath buffer is dispatched to the visibility tester for the computation of the intersection buffer. The visibility tester instructs the ray generator to produce the rays between the current vertices and their successors, i.e. *the inter-level rays*. The sampler traces the inter-level rays and determines the closest intersections. The edges with new intersections correspond to invalid subpaths. The closest intersection along a subpath edge, other than the original successor, is the anchor of the invalid subpath (chapter 3, equation 3.23). The sampler brings the intersection buffer on the host, computes the anchors and substitutes the old successors with the anchors. Each anchor has the same sampling probability and throughput as the replaced successor (chapter 3, equations 3.24 and 3.27). Each anchor level is stored with the invalid subpath, to ensure that the latter is correctly reconstructed. An anchor level is simply the index of an anchor in the invalid subpath. The anchor levels are also used to gather statistical data about the reconstruction process. This data is provided for each test inspected in the next chapter.

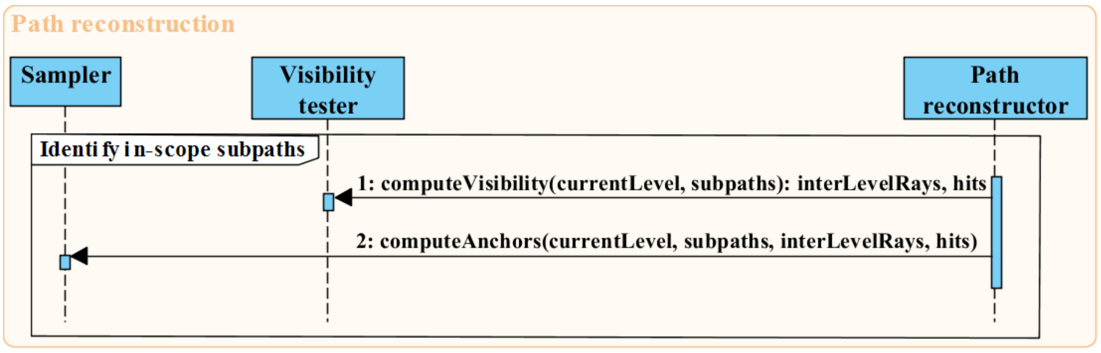

Path reconstruction
Sampler
Visibility tester
Path reconstructor
Identify in-scope subpaths
1: computeVisibility(currentLevel, subpaths): interLevelRays, hits
2: computeAnchors(currentLevel, subpaths, interLevelRays, hits)

Figure 4.9: *The identification of the in-scope subpaths & the computation of anchors.*

Special care must be taken when the inter-level rays intersect the light sources, yielding $S0$ subpaths. An anchor located on a light source requires that all the vertices succeeding the anchor be discarded. $S0$ subpaths are computationally less expensive than the rest of the invalid subpaths, as they no longer need to be reconstructed from the anchor. Figure 4.9 summarizes the identification of the in-scope invalid subpaths.

The out-of-scope identification routine (Figure 4.8, step 2) avoids the exhaustive testing of subpath edges (chapter 3, equation 3.18) by considering only the edges with a dynamic vertex. Given an out-of-scope subpath is disrupted by the movement of one of its vertices, the current identification routine can benefit from this self-deformation by tracing only the edges with a dynamic vertex. Hence, the out-of-scope identification routine begins by selecting only the subpaths that have a dynamic vertex at the current vertex level. The dynamic vertices are identified either through their association with a dynamic object or through the flags that signal the transformation of the camera or of a light source. Each subpath vertex stores the identifier of the primitive that contains it. This identifier is retrieved when the sampler traces the rays and computes their intersections with the geometry. By knowing the dynamic objects, the subpaths with dynamic vertices can be easily identified. The camera transformation flag identifies all the eye subpaths as invalid, since all the roots of the subpaths are altered by the transformation of the camera. On the other hand, the light source transformation flag identifies as invalid only those subpaths that are affiliated with the altered light source.

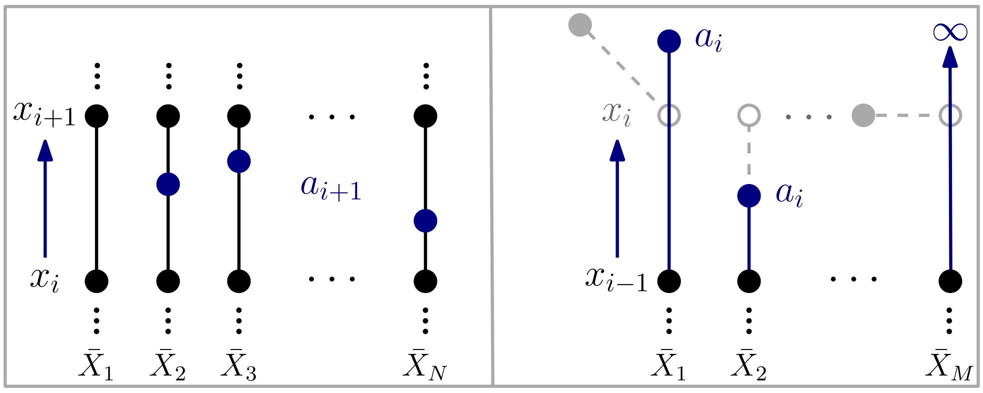


***Figure 4.10**: The identification of the in-scope (left) and of the out-of-scope (right) invalid subpaths for the ith vertex level. The in-scope subpaths are identified by tracing rays between the vertices x_i and x_{i+1}. Only the subpaths $\bar{X}_2$, $\bar{X}_3$ and $\bar{X}_N$ are identified as invalid. The out-of-scope subpaths are identified by tracing rays between the prior vertices x_{i-1} and the old positions (hollow circles) of the dynamic vertices x_i (grey circles). Vertex x_i of the subpath $\bar{X}_2$ moves towards x_{i-1} and yields the anchor a_i. The inter-level ray of the subpath $\bar{X}_M$ fails to produce an anchor by exiting the scene.*

For the subpaths invalidated by the transformation of their roots, new primary rays are sampled (chapter 3, subsection 3.4.1). The out-of-scope subpaths are identified by tracing rays between the previous level of vertices and the current level of dynamic vertices (chapter 3, equation 3.23). The inter-level ray of a dynamic root cannot be generated using this approach, as there is no preceding vertex and the old direction may be incongruous with the new camera/light source configuration. The other vertex levels are processed by tracing rays between the predecessors and the dynamic vertices.

The anchors of the out-of-scope subpaths are computed just as for the in-scope subpaths. The out-of-scope and the in-scope identification routines differ only in that the former routine processes exclusively subpaths with dynamic vertices, whereas the latter routine processes all subpaths. Hence, the out-of-scope routine first selects the subpaths with dynamic vertices and then executes the visibility (Figure 4.9, step 1) and the anchor computations (Figure 4.9, step 2). The in-scope and the out-of-scope identification routines execute one step of the identification process (Figure 4.8). That is, they process only one vertex level across multiple subpaths. The reconstruction process continues until all the vertex levels of the subpaths are exhausted. Figure 4.10 illustrates the identification and the anchor computation for both types of invalidation.

4.3.2 Intra-subpath reconnection

After one identification step, the reconstruction process reconnects as many in-scope and out-of-scope subpaths as possible using the two-chain scenario. The computation of the anchors generated the first chains of the invalid subpaths. The first chain subsumes all the vertices up to the anchor, whereas the second chain includes all the vertices downstream from the anchor. The throughput of the first chain is immutable (chapter 3, equation 3.27) and it is stored at every vertex during subpath generation.

The path reconstructor must reconnect the first and the second chains of the invalid subpaths. The necessary condition for an intra-subpath connection is to find a second chain vertex visible from the anchor (chapter 3, equation 3.29). The reconnection routine uses the inter-level visibility routine (Figure 4.9, step 1) to generate the shadow rays between the anchors and their successors and to compute the intersection buffer.

For each visible ray, the reconnection routine computes the sufficient condition (chapter 3, equation 3.38). First, the tentative connection contribution is computed by considering the connecting edge both as an outgoing and as an incoming direction. Then, the tentative connection contribution is passed through a Russian roulette test (chapter 3, equation 3.31) to randomly discard any tentative connection with a smaller contribution than a given threshold $\varkappa$. The probability for scattering to occur, in and from the direction of the connecting edge, is also computed. The stochastic viability of the tentative connection is determined by means of another Russian roulette test (chapter 3, equation 3.36). The reconnection routine combines these two rejection tests to determine whether the tentative intra-subpath connection can be established.

If the intra-subpath connection succeeds, the radiance and the probabilities of the first two vertices ensuing the anchor are updated using the data computed for the sufficient condition. The anchor level is paired with the welding level, i.e. with the index of the immediate successor. The pair of anchor-welding levels is stored in the subpath. Then, the throughput of the second chain is re-evaluated (chapter 3, equations 3.40 & 3.42).

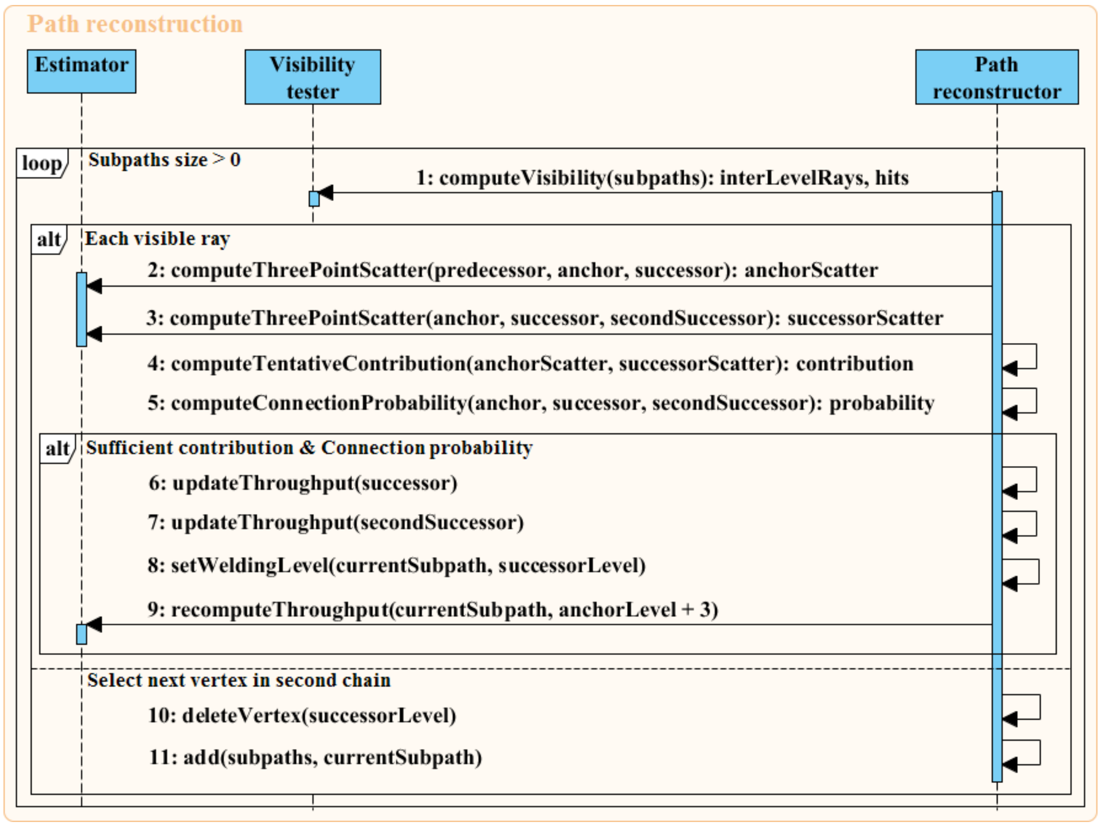

Path reconstruction
Estimator
Visibility tester
Path reconstructor
loop
Subpaths size > 0
1: computeVisibility(subpaths): interLevelRays, hits
alt
Each visible ray
2: computeThreePointScatter(predecessor, anchor, successor): anchorScatter
3: computeThreePointScatter(anchor, successor, secondSuccessor): successorScatter
4: computeTentativeContribution(anchorScatter, successorScatter): contribution
5: computeConnectionProbability(anchor, successor, secondSuccessor): probability
alt
Sufficient contribution & Connection probability
6: updateThroughput(successor)
7: updateThroughput(secondSuccessor)
8: setWeldingLevel(currentSubpath, successorLevel)
9: recomputeThroughput(currentSubpath, anchorLevel + 3)
Select next vertex in second chain
10: deleteVertex(successorLevel)
11: add(subpaths, currentSubpath)

***Figure 4.11**: The reconnection routine with the new intra-subpath connectivity strategy.*

If the intra-subpath connection fails and the second chain still has unprocessed vertices, then the reconnection routine selects the next vertex in the second chain. The intra-subpath connection attempt is resumed with the new pair of connecting vertices. The reconnection routine executes until all the invalid subpaths are reconnected or the vertices of their second chains are exhausted. Figure 4.11 shows the reconnection loop.

Like the identification methods, the reconnection routine processes a single vertex level at a time. The reconstruction routine executes until all the vertex levels of the subpaths are processed. This ensures that subpaths are adequately reconstructed even if they are invalidated several times. Multiply invalidated subpaths store as many reconstruction levels as the number of invalidation instances. A reconstruction level simply denotes an anchor-welding pair. When the reconstruction routine concludes, all the invalid subpaths are collected and dispatched to the regeneration routine. Figure 4.12 includes the reconnection routine in the identification loop (Figure 4.8) and yields the reconstruction routine. Figure 4.13 illustrates the reconstruction of invalid subpaths.

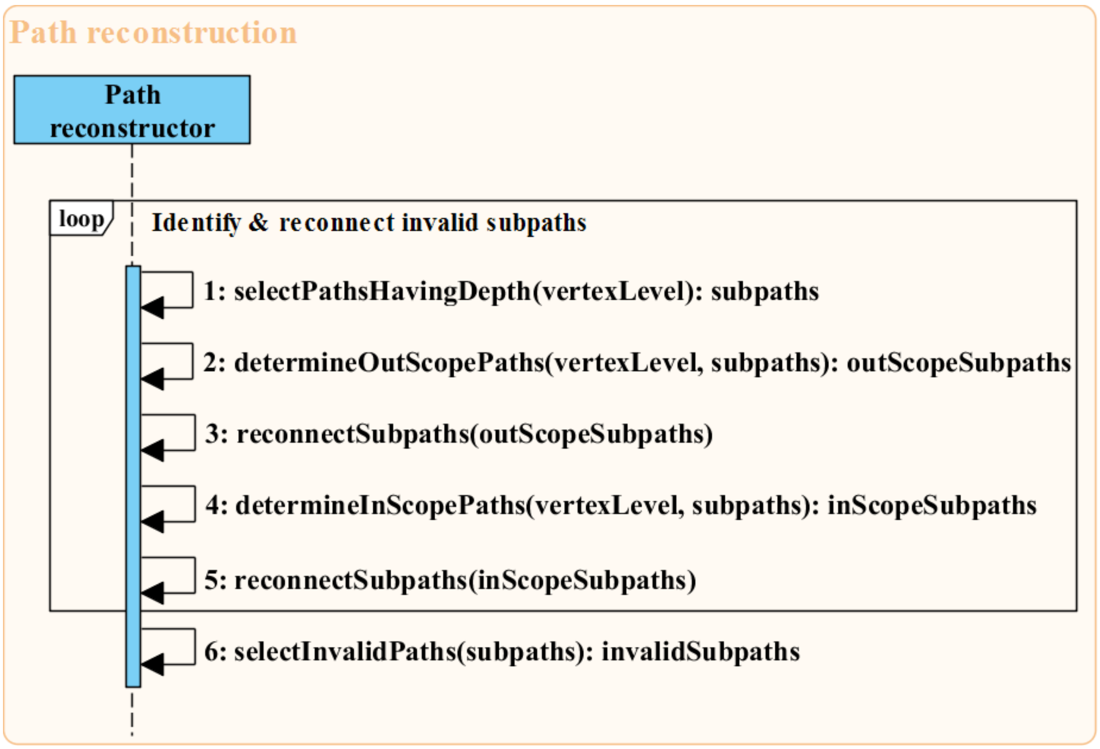

Path reconstruction
Path reconstructor
loop
Identify & reconnect invalid subpaths
1: selectPathsHavingDepth(vertexLevel): subpaths
2: determineOutScopePaths(vertexLevel, subpaths): outScopeSubpaths
3: reconnectSubpaths(outScopeSubpaths)
4: determineInScopePaths(vertexLevel, subpaths): inScopeSubpaths
5: reconnectSubpaths(inScopeSubpaths)
6: selectInvalidPaths(subpaths): invalidSubpaths

Figure 4.12: *The progressive reconstruction of in-scope and of out-of-scope subpaths.*

4.3.3 Subpath regeneration

The reconstruction routine concludes by collecting all the invalid subpaths. The resultant buffer also includes subpaths that either have subminimal lengths or could not be reconnected through intra-subpath connections. In both cases, the subpaths need to be regenerated from their last vertices. The subpath regeneration is performed via the standard local path sampling techniques discussed in chapter 3 (subsection 3.1.1).

The regeneration routine begins by selecting the subminimal and the unconnected subpaths. A *subminimal subpath* is a subpath that does not intersect any light source and has a shorter length than the minimum subpath length (subsection 4.2.3). An *unconnected subpath* is a subpath that has a failed intra-subpath connection and a last anchor level paired with a welding level of -1. The subminimal subpaths are directly stored in the buffer that will be sent to the path generation routine. The unconnected subpaths are first subjected to the continuation test (subsection 4.2.3) and only the ones which pass the test and can be further extended are inserted in the regeneration buffer.

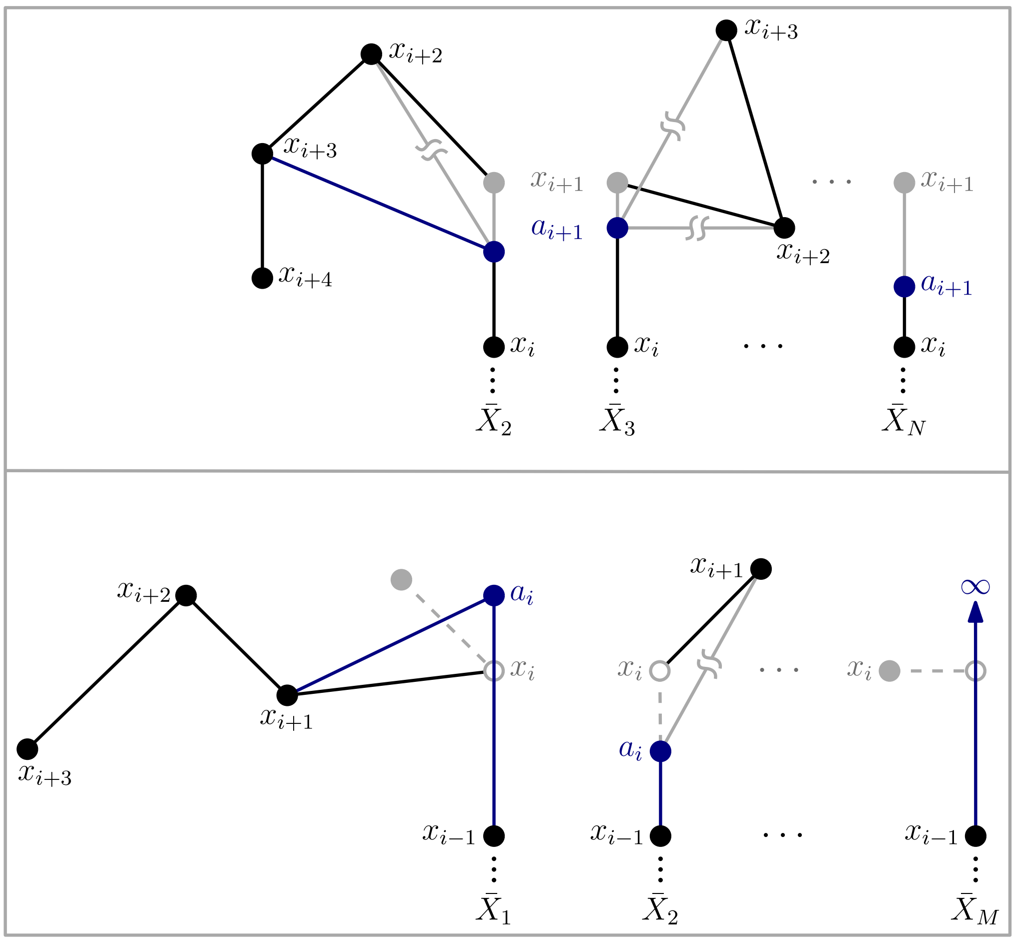


Figure 4.13*: The reconstruction of the in-scope (top) and of the out-of-scope (bottom) subpaths. For the in-scope subpath $\bar{X}_2$ the intra-subpath connection fails between vertices $a_{i+1} - x_{i+2}$ but succeeds at vertex x_{i+3}. The anchor a_{i+1} of the subpath $\bar{X}_3$ cannot be reconnected to any ensuing vertex, whereas for the subpath $\bar{X}_N$ it has no other vertices to be connected to. The in-scope subpaths $\bar{X}_3$ and $\bar{X}_N$ will be regenerated from the anchor. For the out-of-scope subpath $\bar{X}_1$ the intra-subpath connection succeeds at the first successor x_{i+1}, whereas for the subpath $\bar{X}_2$ it fails. The inter-level ray of the subpath $\bar{X}_M$ exits the scene without generating an anchor. The out-of-scope subpaths $\bar{X}_2$ and $\bar{X}_M$ will be regenerated from vertex a_i, respectively from vertex x_{i-1}.*

The minimum subpath length is used to avoid any extra variance arising from short subpaths. By applying the continuation test, the invalid subpaths are appropriately regenerated. No subminimal subpath is constrained to have a shorter length than the specified minimum, just as no unconnected subpath is extended more than necessary.

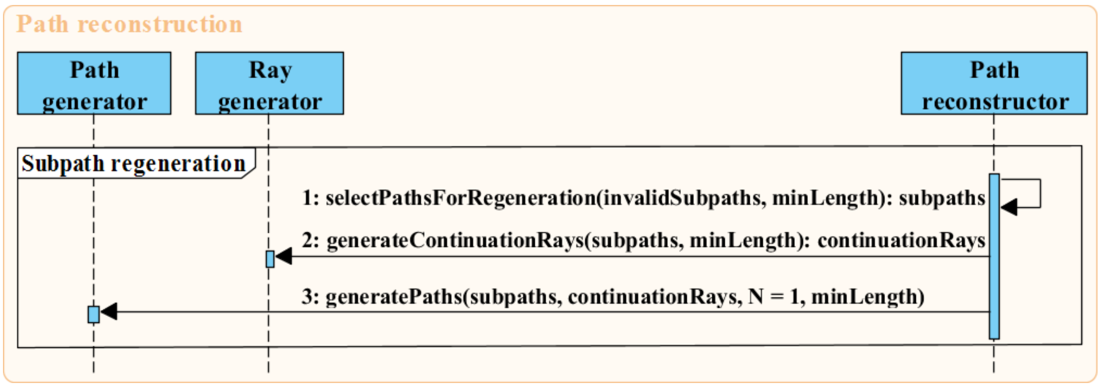

Path reconstruction
Path generator
Ray generator
Path reconstructor
Subpath regeneration
1: selectPathsForRegeneration(invalidSubpaths, minLength): subpaths
2: generateContinuationRays(subpaths, minLength): continuationRays
3: generatePaths(subpaths, continuationRays, N = 1, minLength)

***Figure 4.14**: The joint regeneration process for subminimal & unconnected subpaths.*

The selection routine (Figure 4.14, step 1) suppresses the use of Russian roulette on subminimal subpaths and dispatches the latter to the standard path generation routine. Should Russian roulette be applied sooner than actually required, the lower energy transported by the subminimal subpaths will cause darker regions to appear in the current frame. Similarly, unconnected subpaths extended more than necessary, due to the suppression of Russian roulette at their last anchors, will carry more energy than the rest of the subpaths and will cause brighter regions to appear in the frame. Consequently, the selection routine is pivotal to the correct regeneration of subpaths.

Next, the regeneration routine instructs the ray generator to produce the continuation rays for the selected subpaths (Figure 4.14, step 2). For the unconnected subpaths, the continuation rays are generated from the last anchors that could not be reconnected via the intra-subpath connectivity strategy. For the subminimal subpaths, the continuation rays are generated from their last vertices. All the continuation rays are generated by sampling the local probability density functions defined at the last subpath vertices.

Using the selected subpaths and the continuation rays, the regeneration routine invokes the path generator to extend the subpaths (Figure 4.14, step 3). Once the path generation loop completed, the regenerated and the reconstructed subpaths are merged in a single buffer. By invoking the path generator (Figure 4.2), the path reconstructor seamlessly re-enters the standard Monte Carlo pipeline. This interoperability is the advantage of the modular design. Figure 4.14 outlines the steps executed by the regeneration routine.

4.3.4 Contribution re-evaluation

The regeneration routine re-enters the standard Monte Carlo pipeline by invoking the path generation routine with the subminimal and unconnected subpaths (Figure 4.14, step 3). Still, the reconstructed subpaths can be reused in bidirectional connections only after the estimator computes their geometric factors, reverse probabilities and partial MIS weights. Once this information is computed, the reconstructed and the valid light subpaths are merged and the resultant buffer is used to randomly assign light subpaths to each vertex of the reconstructed eye subpaths. The contributions of the full light transport paths are evaluated normally, as delineated in subsection 4.2.5.

When the camera is altered, all the eye subpaths are reconstructed and re-evaluated. In the case of a light source transformation, only the affiliated light subpaths and the subpaths disrupted by the geometry of the light source are reconstructed. The latter category of subpaths also includes light subpaths (besides eye subpaths), if there are multiple light sources present in the scene. However, the contributions of all the eye subpaths that were connected to the altered light source and to its affiliated light subpaths change implicitly. Thus, all the eye subpaths with disrupted geometry or with invalid contributions are re-evaluated. The subpaths with intact geometric structures, yet with invalid contributions, are designated as collateral. If the only light source of the scene was transformed, then the entire buffer of eye subpaths is re-evaluated. Otherwise, only the disrupted and the collateral eye subpaths are re-evaluated. The collateral eye subpaths are identified through a field that associates each eye vertex with a random light subpath. All the eye subpaths that have a vertex associated with a reconstructed light subpath are re-evaluated through the process depicted in Figure 4.6.

The transformation of an object that is neither the camera nor a light source, does not require the re-evaluation of the full buffer of eye subpaths. Neither does the alteration of a light source when there are multiple light sources in the scene. The next subsection discusses the identification and re-evaluation of collateral subpaths for such changes.

4.3.5 Collateral subpath identification and re-evaluation

The path reconstructor currently implements two methods for identifying the collateral subpaths. The first method identifies the collateral subpaths by parsing the eye subpath buffer and selecting the subpaths that have a vertex within the surrounding perimeter of a transformed object. Currently, the perimeter comprises two extents. The first extent is computed before the transformation of the object and is as wide as the size of the bounding box determined at that moment. The second extent starts from the altered object and is as wide as its bounding box. Both extents are computed in the same way, but at different moments in the lifespan of a dynamic object. Thus, any eye subpath with a vertex either in the first or in the second extent is identified as collateral.

However, this method is limited to simple transformations and is not effective when more complex transformations are applied to the scene. For this reason, the second method uses both a shadowing identifier and the association between an eye vertex and a light subpath to select the collateral subpaths. Each eye vertex stores the identifier of its allotted light subpath. All the eye subpaths which have a vertex associated with a reconstructed light subpath, are identified as collateral. Furthermore, each subpath stores a shadowing identifier, which is set to the identifier of the primitive that first obstructs a connection between the given eye subpath and its allotted light subpaths. The estimator connects each eye subpath vertex to a random light subpath (Figure 4.6). As soon as the estimator detects an obstructed connection between an eye subpath vertex and a light subpath, it will set the shadowing identifier to the identifier of the occluding primitive. If a subpath already has its shadowing identifier set and another occlusion is detected, the estimator will not set the identifier to the currently occluding primitive. Like this, the lower vertex levels are prioritized over the higher ones, which usually contribute less to the image. All the subpaths which have a nonnegative shadowing identifier, are identified as collateral. This method returns more collateral subpaths than the first one, as it considers both subpaths with invalid contributions and subpaths from the shadow areas cast by the static objects. Yet, it is more efficient at identifying the collateral subpaths for generic scene transformations. Regardless of the identification method, the collateral subpaths are randomly connected to light subpaths and the resultant paths are evaluated through the process described in subsection 4.2.5.

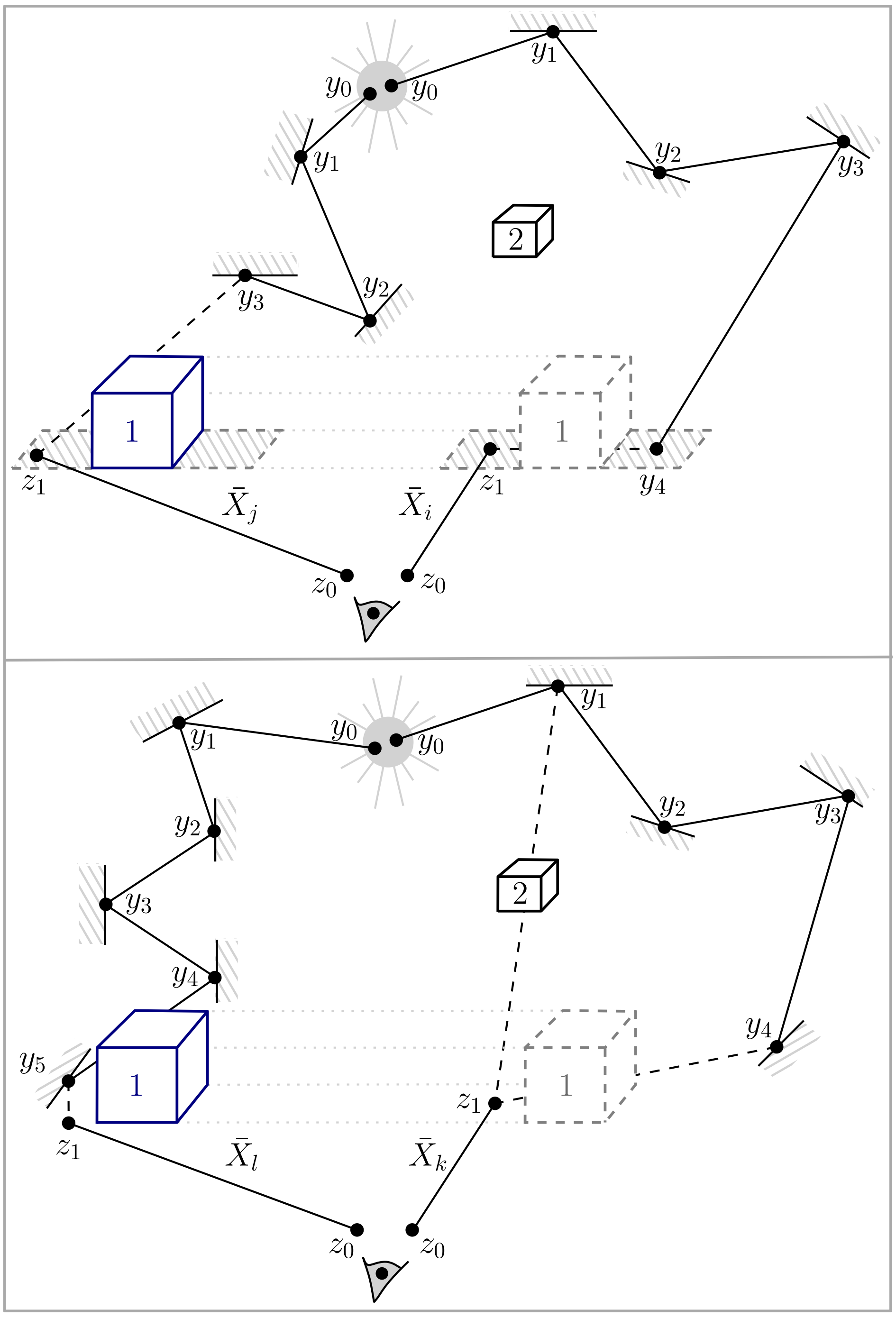


***Figure 4.15**: The identification of collateral subpaths using the perimeter method (top) and the shadowing identifier and light subpath association method (bottom). Both eye subpaths $\bar{X}_i$ and $\bar{X}_j$ are identified as collateral since $\bar{X}_i$ has a vertex within the extent (dashed grey zones) of the object before transformation, while $\bar{X}_j$ has a vertex within the extent of the altered object (navy). Eye subpath $\bar{X}_k$ stores as shadowing identifier the index of the first intersected primitive belonging to object 2. However, it does not store any identifier associated with object 1, since the occlusion occurs at a higher light vertex level and the shadowing identifier is already set. The light subpath allotted to vertex z_1 of the eye subpath $\bar{X}_l$ is invalidated and will be reconstructed, causing $\bar{X}_l$ to be identified as collateral, even though the connection $y_5 - z_1$ is not occluded. The collateral subpath identification is similar for higher vertex levels of the eye subpaths.*

Figure 4.15 exemplifies the identification of collateral subpaths using the two methods. Neither method is optimal, in the sense that all the vertices of the collateral subpaths are reconnected to newly selected light subpaths, even if their contributions are valid. Multiple paths are generated from the same eye subpath, by connecting each of its vertices to all the vertices of a randomly assigned light subpath. An eye subpath is identified as collateral based on the properties of one of its vertices. Yet, the evaluation routine establishes bidirectional connections not just for one vertex of the collateral subpath, but for all its vertices. This yields redundant computations, as the connections between some of the eye vertices and their light subpaths are still valid. Ideally, only the eye vertices with affected connections should be re-evaluated. Despite being more time-consuming, the exhaustive re-evaluation of collateral subpaths is not detrimental.

The re-evaluation of the collateral subpaths concludes the evaluation phase. Before the current frame is serialized and composited with the prior frames, the reconstruction levels of each subpath are cleared. The flushing of the anchor-welding pairs ensures that the reconstruction process does not leak between frames. That is, the subpaths reconstructed in a previous frame are reprocessed only if they are invalid for the current frame. The image synthesis phase concludes the entire reconstruction process.

4.3.6 The path manipulation algorithm

The path manipulation algorithm supports dynamism by extending bidirectional path tracing to reuse paths in the temporal domain. The generated subpaths are reconstructed such that the coherence of the light flow is preserved between rendering frames. By reconstructing and reusing paths, the path manipulation algorithm supplants the static usage of paths and of sampling techniques with a generation-evaluation-reuse cycle.

The path manipulation algorithm starts by identifying the invalid subpaths. The identification of the invalid subpaths (Figure 4.8) is performed based on the type of subpath deformation. The in-scope subpaths are identified by searching for obstructed edges, whereas the out-of-scope subpaths are identified by searching for edges with dynamic vertices. The in-scope and the out-of-scope routines compute the anchors of all the identified subpaths. The anchor computations effectively create the first chains of the invalid subpaths. The reconnection routine (Figure 4.11) uses the intra-subpath connectivity strategy to establish all the possible connections between the first and the second chains of the invalid subpaths. When all the vertex levels in the subpath buffer have been processed, the reconstruction routine (Figure 4.12) collects and dispatches all the invalid subpaths to the regeneration routine (Figure 4.14). The regeneration routine appropriately selects and extends the subminimal and unconnected subpaths. The regenerated and the reconstructed subpaths are merged in a single buffer. The estimator computes the geometric factors, the reverse probabilities and the partial MIS weights for each subpath in the given buffer. The light subpaths are processed independently of the eye subpaths. The reconstructed and the valid light subpaths are merged and the resultant buffer is used to assign a random light subpath to each vertex of the reconstructed eye subpaths. The same buffer is used to assign random light subpaths to the identified, collateral subpaths (subsection 4.3.5). Then, the estimator evaluates the contributions of all the resultant paths (Figure 4.6). Finally, the camera constructs the current frame, composites it with the prior frames and serializes it. If the path manipulation algorithm must execute additional iterations, the pipeline will resume with the identification of the invalid subpaths, otherwise it will serialize the composite frame. Figure 4.16 illustrates the schematic flow of the path manipulation algorithm. Note that the reconstruction pipeline replaces only the path generation phase.

4.4 Conclusions

This chapter discussed the implementation of the light transport framework. The focal characteristic of the light transport framework is modularity. The building blocks that compose the framework, define both the standard Monte Carlo algorithms (section 4.1) and the path manipulation algorithm (subsection 4.3.6). The standard algorithms entail different degrees of generality and they are implemented by varying the functionality of the blocks that define the standard Monte Carlo pipeline (section 4.2). The path manipulation algorithm is defined by the reconstruction pipeline, which is simply an adaptation of the standard Monte Carlo pipeline to the path manipulation strategies.

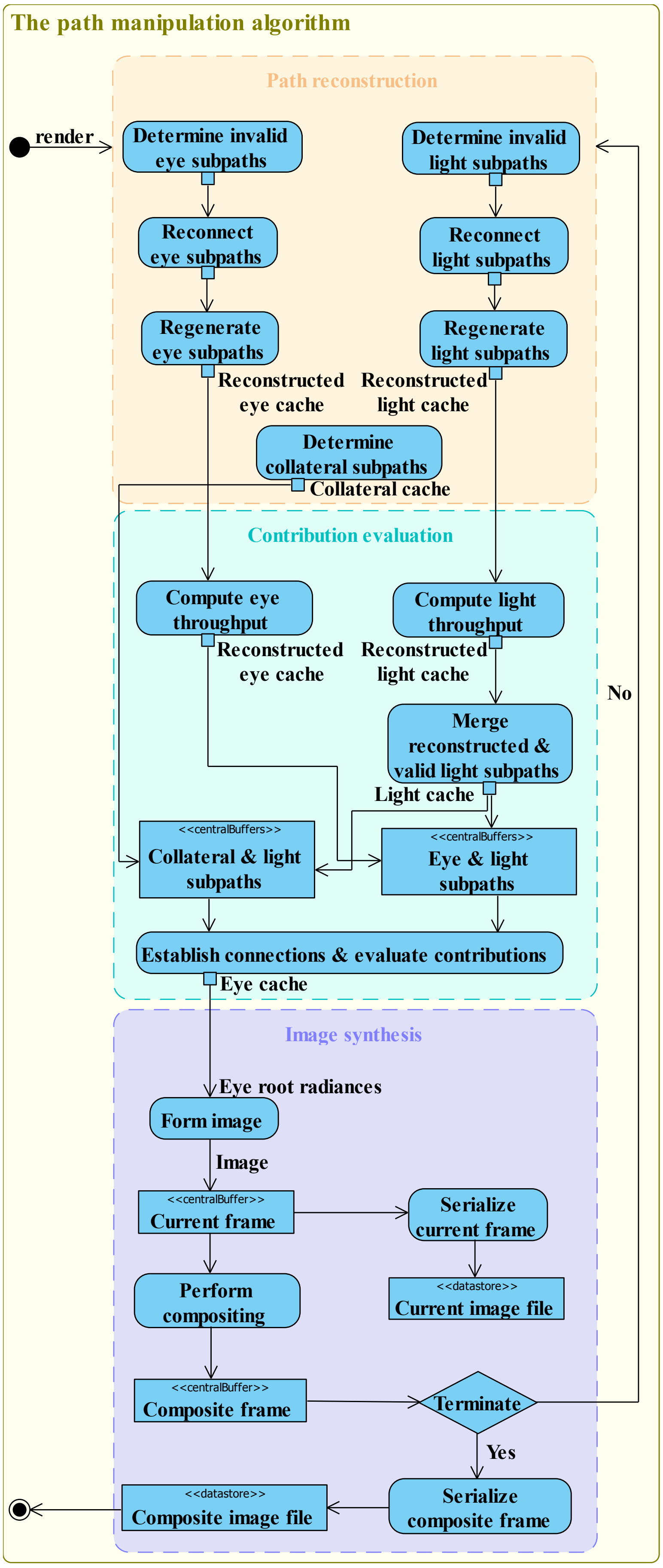

The path manipulation algorithm
Path reconstruction
render
Determine invalid eye subpaths
Determine invalid light subpaths
Reconnect eye subpaths
Reconnect light subpaths
Regenerate eye subpaths
Regenerate light subpaths
Reconstructed eye cache
Reconstructed light cache
Determine collateral subpaths
Collateral cache
Contribution evaluation
Compute eye throughput
Compute light throughput
Reconstructed eye cache
Reconstructed light cache
No
Merge reconstructed & valid light subpaths
Light cache
<<centralBuffers>> Collateral & light subpaths
<<centralBuffers>> Eye & light subpaths
Establish connections & evaluate contributions
Eye cache
Image synthesis
Eye root radiances
Form image
Image
<<centralBuffer>> Current frame
Serialize current frame
<<datastore>> Current image file
Perform compositing
<<centralBuffer>> Composite frame
Terminate
Yes
Serialize composite frame
<<datastore>> Composite image file

Figure 4.16: *The path manipulation algorithm defined via the path reconstructor block.*

The standard pipeline includes the path generation, contribution evaluation and image synthesis phases. The path generation phase is implemented with the functionality of the ray generator, sampler and path generator blocks (subsections 4.2.1 - 4.2.3). The path generation process reflects the style of the progressive Monte Carlo methods. Subpaths are generated using a similar approach to Novák et al. (2010), except that they are stochastically terminated via the Russian roulette test proposed by Veach (1998, p. 309) and the primary rays of the restarted subpaths are always resampled. The latter decision is particularly beneficial for effects like motion blur and depth of field. The contribution evaluation phase uses the visibility tester and estimator blocks (subsections 4.2.4 - 4.2.5) to construct and assess bidirectional paths. Multiple paths are efficiently generated from the same pair of subpaths (Veach 1998, p. 300), with the advantages of zero costs and reduced variance. However, each eye subpath vertex is connected to a random light subpath (Novák et al. 2010), to reduce the deterministic noise and path correlation. Two methods (Veach 1998; van Antwerpen 2011b) are used to compute the multiple importance sampling weights that reduce variance. Finally, the image synthesis phase uses the camera (subsection 4.2.6) to produce the frames.

The above state-of-the-art techniques improve both accuracy and performance. Still for a similar image quality, the path manipulation algorithm outperforms bidirectional path tracing (subsection 4.2.7). The performance and statistical properties of the path manipulation algorithm are analysed in chapter 5. The reconstruction pipeline, which defines the proposed algorithm, replaces only the path generation phase of the standard pipeline. Instead of regenerating the entire collection of paths, the path reconstructor retains and adapts the paths to the new scene configuration (subsections 4.3.1 - 4.3.3). After identifying and reconnecting the invalid subpaths, the reconstructor seamlessly re-enters the standard pipeline. Thus, the reconstructed and the collateral subpaths (subsection 4.3.5) are re-evaluated through the standard process. By reconstructing and reusing paths, the path manipulation algorithm extends bidirectional path tracing to the temporal domain and supports scene dynamism. Still, the path manipulation strategies reconstruct paths regardless of the subpath or scene dynamism type. Hence, they can extend most path reliant algorithms to reuse paths in the temporal domain.

Chapter 5

Results

This chapter analyses the results of the path manipulation algorithm. By reconstructing and reusing paths, the path manipulation algorithm extends bidirectional path tracing to the temporal domain and supports scene dynamism for high-quality light transport simulations. The next sections evince its performance over bidirectional path tracing.

Five tests are detailed in this chapter. The first three execute affine transformations inside the Cornell box. As a rendering benchmark, the Cornell box enables the easy identification of illumination effects/artefacts and thus it is used to ensure the path manipulation algorithm satisfactorily conforms to this standard. These tests cover the major dynamism types by transforming an object (section 5.1), the light source (section 5.2) or the camera (section 5.3). As an influencing factor of the intra-subpath connectivity, the number of transformation units is specified for each test. Subsection 5.1.3 discusses the factors that influence the intra-subpath connectivity. Devised for product assembling/disassembling sequences in CAD/CAM systems, the fourth test applies affine transformations to various body parts. Its rationale is to emulate a typical project undertaken by Optis and to emphasize the flexibility of the proposed algorithm in operating without predefined animation paths. By transforming multiple objects, this test lends itself to scalability analysis. The fifth test emulates another Optis project and evinces the generality of the reconstruction process by combining object, light source and camera transformations. Both tests are important to various industries, like the automotive and architectural sectors that Optis serves, as they exhibit dynamism with increased performance. All five tests examine the information gathered from the reconstruction process and use a breakdown of the execution time to inspect the performance gap between bidirectional path tracing and the path manipulation algorithm. The exposition also addresses the intra-subpath connection viability. The results were produced using a 24GB memory, a 2.6 GHz Intel Core i7 processor with 4 cores and an NVIDIA GeForce GTX 980M graphics card with 1536 CUDA cores.

Configuration	**Scenes**		
	Cornell box	*Motorbike*	*Bathroom*
Image resolution	512×512	720×486	640×512
Light sources	1	3	24
Triangles	48	117,794	1,120,562
Samples per light source	5000	1000	200
Samples per pixel	100/100/90	100	80
Samples per pixel in reference image	10,000	40,100	10,240
Intel Core i7 processor $\begin{cases} cores = 4 \\ frequency = 2.6\ GHz \end{cases}$			
NVIDIA GeForce GTX 980M $\begin{cases} CUDA\ cores = 1536 \\ graphics\ clock = 1038\ MHz \end{cases}$			

***Table 5.1**: Test configuration for the rendered scenes. In the case of the Cornell box 100 samples per pixel are generated for the translation of the cube and of the light source, whereas 90 samples per pixel are generated for the rotation of the camera.*

Table 5.1 enumerates the parameters used to configure all of the discussed tests. A subpath stores its identifier, a termination flag, the intersected light source identifier, the shadowing identifier, its affiliation, the reconstruction levels and its vertices. These members, except the boolean termination flag and vertices, are coded using integers. Assuming the size of both a boolean and an integer is 4 bytes, the first five members require 24 bytes of memory. Each subpath vertex stores the containing triangle identifier, the affiliated light subpath identifier, an intersection value, a surface point, the radiance/importance, the forward and reverse probabilities, the geometric factor and the partial MIS weights. The integer identifiers with the other floating-point based members amount to 60 bytes. By conservatively bounding the reconstruction levels to the subpath length, the memory per average length subpath can be estimated via:

$$\varsigma_{SP} = 24bytes + \underbrace{(N_\tau - 1) \cdot 8bytes}_{reconstruction\ levels} + \underbrace{N_\tau \cdot 60bytes}_{subpath\ vertices} \tag{5.1}$$

where N_τ is the average number of vertices per subpath. The subpath buffer also stores the number of samples (1 integer) per affiliation (2 integers). The memory required by subpaths per test can be estimated using their reported number and average length via:

$$\varsigma_{\varphi_i} = \underbrace{(N_{EP} + N_{LP})}_{total\ subpaths} \varsigma_{SP} + \underbrace{\left(N_{pixels} + N_{lights}\right)}_{total\ affiliations} \cdot 12bytes \tag{5.2}$$

where $N_{EP/LP}$ is the number of eye/light subpaths. Chapter 6 reports the total memory used by the path manipulation algorithm per scene. The reported memory regards any amount of memory used throughout execution, not just that allocated for the subpaths.

5.1 Cornell box: geometry transformation

One of the experiments used to analyse the path manipulation algorithm refers to rendering ten 512×512 frames of the Cornell box, with a white cube being translated 2 units on the X axis. The first frame of this animation sequence is rendered with the bidirectional path tracing algorithm implemented in the light transport framework. The rest of the animation frames are generated with the path manipulation algorithm, which reconstructs and reuses the existing paths. Figure 5.1 illustrates three instances of the moving cube, as it progresses from the centre of the scene towards the green wall. The frames were produced using 100 samples per pixel and 5000 samples per light source.

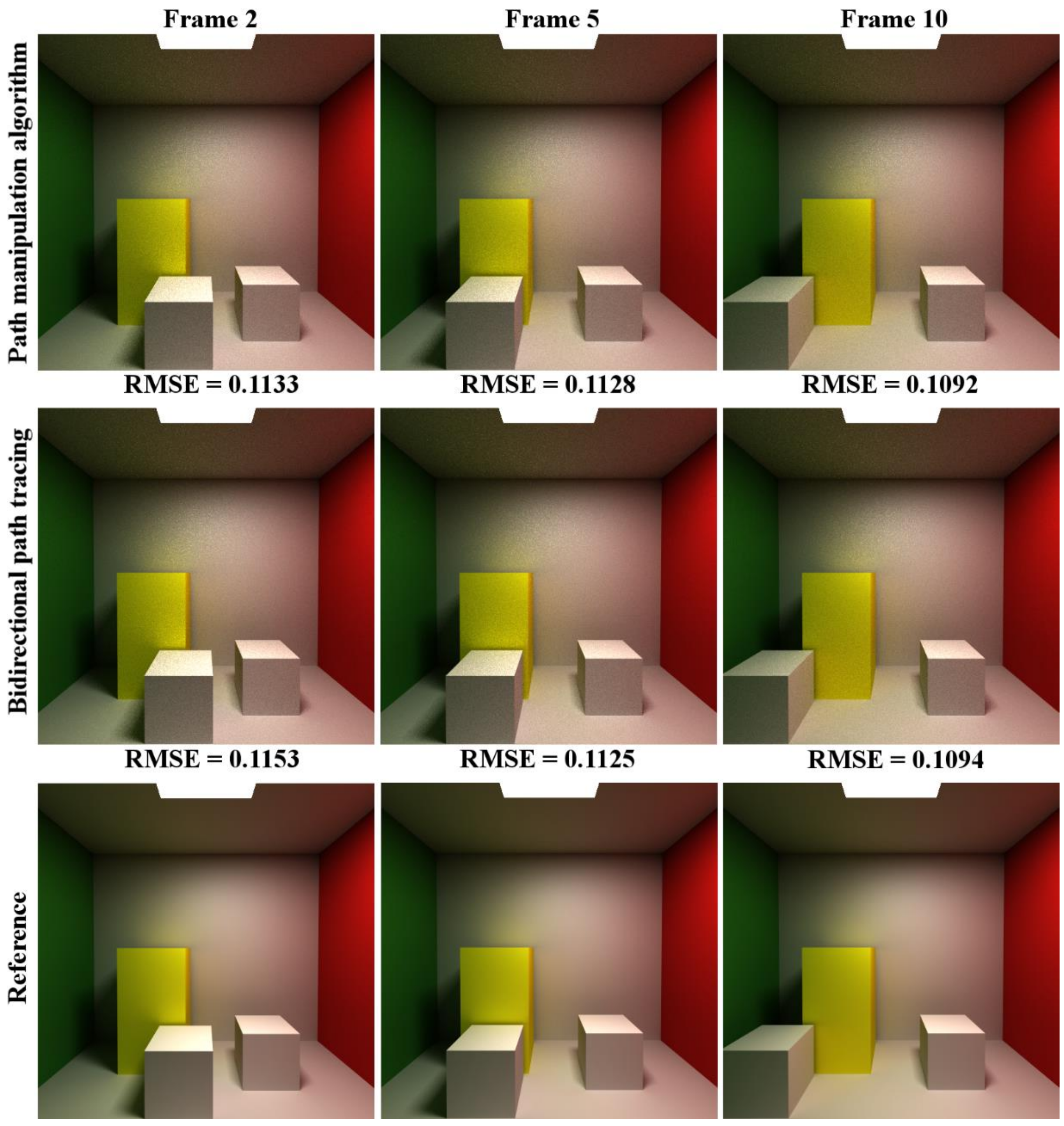


Figure 5.1*: Comparison between the path manipulation algorithm and bidirectional path tracing using an equal number of samples per pixel, respectively per light source.*

Algorithm / Steps	Bidirectional path tracing			Path manipulation		
	Frame 2	*Frame 5*	*Frame 10*	*Frame 2*	*Frame 5*	*Frame 10*
EPs generation	621.908s	675.491s	714.087s	—	—	—
LPs generation	112.920s	99.641s	100.243s	—	—	—
Invalid EPs reconstruction	—	—	—	109.584s	105.146s	107.359s
Invalid LPs reconstruction	—	—	—	0.282s	0.297s	0.297s
EPs regeneration	—	—	—	24.209s	22.67s	27.112s
LPs regeneration	—	—	—	0.031s	0.078s	0.047s
Collateral EPs identification	—	—	—	17.877s	6.532s	24.972s
EPs throughput evaluation	84.549s	35.785s	57.956s	11.517s	9.251s	12.944s
LPs throughput evaluation	0.031s	0s	0.031s	0s	0s	0s
Bidirectional contribution evaluation	985.224s	1074.22s	1137.29s	492.907s	437.011s	449.887s
Total execution time	**1804.79s**	**1885.56s**	**2009.86s**	**657.98s**	**582.11s**	**625.149s**

***Table 5.2**: Breakdown of the execution time employed by bidirectional path tracing, respectively by the path manipulation algorithm in generating the 2^{nd}, 5^{th} and 10^{th} animation frame. EPs denotes the eye subpaths, while LPs denotes the light subpaths.*

The reference frames were generated with bidirectional path tracing using 10,000 samples per pixel. The root mean squared errors, reported throughout this chapter, were computed by taking the square root from the relative mean squared errors. The relative mean squared error, for a given image, was defined as the mean over all pixel errors. The error of a pixel was computed as $\sum_i \left(I_j^i - R_j^i\right)^2 / (\bar{R}_j^2 + \epsilon)$, where i sums the squared difference between the evaluated and reference pixels over the colour channels, $\bar{R}_j$ is the mean of all colour channels of the j^{th} reference pixel and $\epsilon = 0.001$. The relative mean squared error formula is the one proposed by Bauszat et al. (2017).

5.1.1 Gain analysis

The comparable root mean squared errors, associated with the frames depicted in Figure 5.1, corroborate the assertion that the path manipulation algorithm does not enhance bidirectional path tracing with additional sampling strategies (chapter 2, subsection 2.6.1). For a given number of samples, the proposed algorithm attains a similar image quality in a reduced amount of time. Table 5.2 details the performance gain of the path manipulation algorithm over the baseline. A considerable reduction in the execution time occurs between the path generation and path reconstruction phases.

The average decrease between the generation of paths (Table 5.2, steps 1-2) and their reconstruction (Table 5.2, steps 3-7) is 80.77%. This performance gap is determined by the operations executed in each phase. The path generation routine (chapter 4, subsection 4.2.3) determines the ray-primitive intersections, appends the computed vertices to the corresponding subpaths, samples the next-bounce rays, determines the subpath lengths and regenerates the primary rays for all the terminated subpaths whose affiliation count does not exceed the specified number of samples. The invalid subpath identification routines (chapter 4, subsection 4.3.1) are more lightweight than the path generation routine, since they only retrace existent rays and compute the anchors. The intra-subpath connectivity strategy (chapter 4, subsection 4.3.2) contributes to the performance improvement by maximizing the path reuse. The reconnection of disrupted subpaths recycles path information, obviates subpath regeneration and reduces the number of edges that need to be retraced. Regeneration is circumvented for all the subpaths which do not have unconnected anchors or subminimal lengths. The reconnection routine reduces the number of traceable edges by discarding the vertices to which an anchor failed to be connected. Hence, the identification routines may benefit from the effect of the reconnection routine for higher-order vertex levels.

The regeneration routine (chapter 4, subsection 4.3.3) invokes the path generation routine with the collection of unconnected and subminimal subpaths, which is usually smaller than the full collection of subpaths. Similarly, the collateral identification routines (chapter 4, subsection 4.3.5) return a moderate number of subpaths. For the current test, the perimeter method was used to identify the collaterally invalid subpaths.

The reconstruction phase directly impacts the contribution evaluation phase. The throughput is evaluated only for the reconstructed light and eye subpaths (Table 5.2, steps 8-9), as the geometric structures of the collateral subpaths are intact and do not require throughput evaluation. Being considerably less than the total number of subpaths, the reconstructed subpaths require an average of only 20.60% from the execution time used to evaluate the throughput of all the generated subpaths. The 10^{th} step detailed in Table 5.2 refers to establishing bidirectional connections between the light and eye subpaths and to evaluating the contributions of the light transport paths.

Bidirectional path tracing evaluates all the generated subpaths, whereas the path manipulation algorithm evaluates only the reconstructed and collateral subpaths. Due to the reduced number of subpaths, the path manipulation algorithm gains an average of 56.57% execution time during the evaluation of the contributions. The total gain of the proposed algorithm over bidirectional path tracing is 67.18% across the three frames.

5.1.2 Reconstruction analysis

Table 5.3 provides the information collected during the reconstruction of the 9 animation frames. Note that the average number of discarded rays amounts to 9.65% from the total number of generated rays. This percentage gives the rays that are not traced by the identification routines, as they are discarded during the reconstruction process. The average number of connected subpaths is approximately 1.18% from the total number of subpaths, whereas the average number of unconnected subpaths represents 15.96% from the total number of subpaths. The intra-subpath connectivity strategy ensures that only subpath chains with sufficiently large contributions are reconnected. The throughput from the anchor onwards must be proportional to the throughput that would be obtained on the subpath, had the latter been generated via the conventional local path sampling techniques. The reconnection routine randomly discards subpath chains with small throughput. Furthermore, subpath chains are not reconnected if the probabilities for scattering to occur, in and from the direction of the connecting edge, fail to pass the Russian roulette test. These two conditions are rather strong and they dictate the connected-unconnected proportion observed in Table 5.3.

Average	**#**	**%**
Invalid primary rays	1,865,139	1.4521
Invalid higher-order rays	2,909,361	2.2651
Discarded rays	12,391,727	9.6480
Total generated rays	128,438,276	100
Out-of-scope subpaths	3,463,425	13.2093
In-scope subpaths	997,044	3.8026
Connected subpaths	308,989	1.1784
Unconnected subpaths	4,185,561	15.9636
Processed subpaths	4,458,372	17.0040
Total generated subpaths	26,219,400	100
Subpath length	4	—
Invalid subpath identification time	60.6799s	54.6709
Reconnection time	47.8354s	43.0984
Total reconstruction time	110.9911s	100

Table 5.3*: The reconstruction information gathered from the 9 frames generated with the path manipulation algorithm. The percentage column in the ray section is computed relative to the total generated rays, while for the subpath section is computed relative to the total generated subpaths. The* 1^{st} *and* 2^{nd} *rows in the time section regard steps 3-4 from Table 5.2 and are computed relative to the total reconstruction time (seconds).*

When an intra-subpath connection fails, the given subpath is regenerated from its unconnected anchor. This means that the figure reported for the unconnected subpaths may include subpaths which are reconnected at lower vertex levels, but contain unconnected anchors at higher levels. The conditions of the intra-subpath connection effectively eliminate unusual patterns, such as brighter or darker regions. Similarly, the use of Russian roulette, in selecting the subpaths that must be regenerated, prevents any artefact from appearing in the reconstructed frames (chapter 4, subsection 4.3.3).

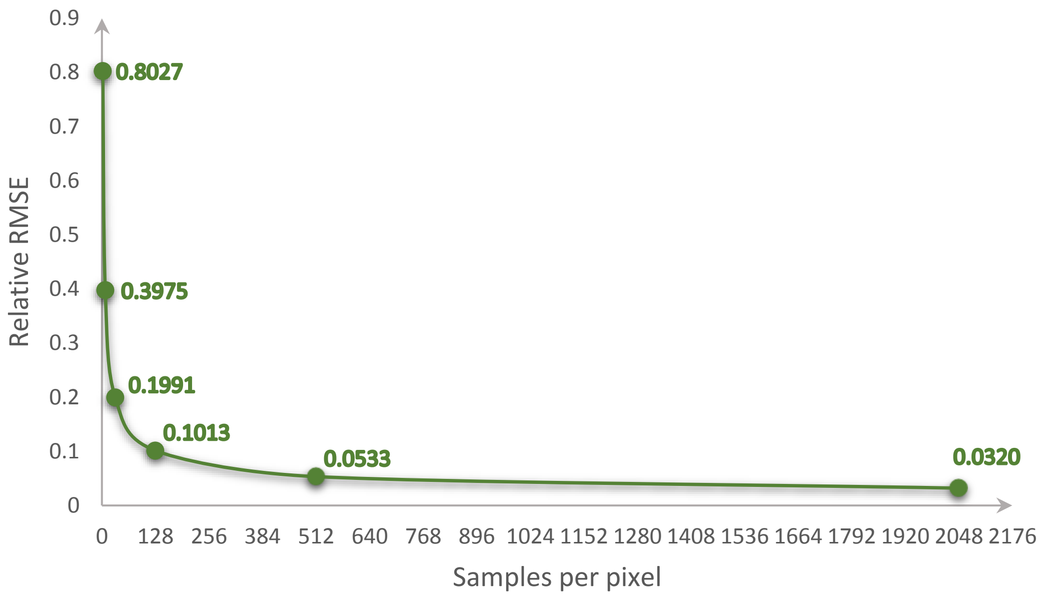


***Figure 5.2**: The convergence behaviour of the path manipulation algorithm exhibited at rendering the* 5^{th} *cube animation frame with* $2, 8, 32, \ldots 2048$ *samples per pixel.*

The number of reconstructed subpaths represents 17% from the number of generated subpaths. As discussed, this percentage has a direct impact on the re-evaluation of the subpath throughput and on the contribution evaluation of the resultant paths. The last 3 rows in Table 5.3 report the time used strictly for the identification and reconnection of invalid subpaths. The last row corresponds to the sum between step 3 and step 4 in Table 5.2, averaged over 9 frames. The previous two rows represent the portion from this average that is used to identify the invalid subpaths, respectively to reconnect them.

Figure 5.2 plots the root mean squared errors computed for the 5^{th} animation frame rendered with the path manipulation algorithm using an increasing number of samples per pixel. The RMSE plots for the other 8 reconstructed frames and for the frames rendered with bidirectional path tracing closely follow the one depicted in Figure 5.2.

5.1.3 Intra-subpath connectivity factors

The test analysed in this section regards the translation of a white cube, 2 units on the X axis. The scattering properties of the dynamic object and the applied transformations affect the intra-subpath connectivity. For example, by altering the colour of the moving cube from white to blue, while maintaining the same brightness, reduces the connected subpaths to 0.58% and increases the unconnected subpaths to 16.45% from the total number of generated subpaths. This test suggests that subpaths with higher energy at the connecting vertices have higher probabilities of being reconnected, which is in conformance with the sufficient throughput condition used to establish intra-subpath connections. Likewise, the closer the connecting edge matches the distributions of the connecting vertices, the higher the probability for the subpath chains to be reconnected.

The transformation type is another factor that influences the intra-subpath connectivity. Closely spaced transformations slightly increase the intra-subpath connectivity, unlike ampler transformations. Table 5.4 shows the connected and unconnected percentages for the white cube and for translations that double the magnitude of their X component. For the analysed tests, the transformation steps were adapted to the scene dimensions.

The intra-subpath connectivity is also a function of the subpath length. Figure 5.3 plots the connectivity percentage and the connected-unconnected ratio, for the original test (2-unit translations of the white cube on the X axis) and for increasing subpath lengths.

Translation units on *X* axis	Connected subpaths	Unconnected subpaths
-0.5	1.4871%	15.1149%
-1.0	1.3281%	15.1923%
-2.0	1.1784%	15.9636%
-4.0	1.0395%	16.8775%

Table 5.4: *The influence of the transformation type on the intra-subpath connectivity.*

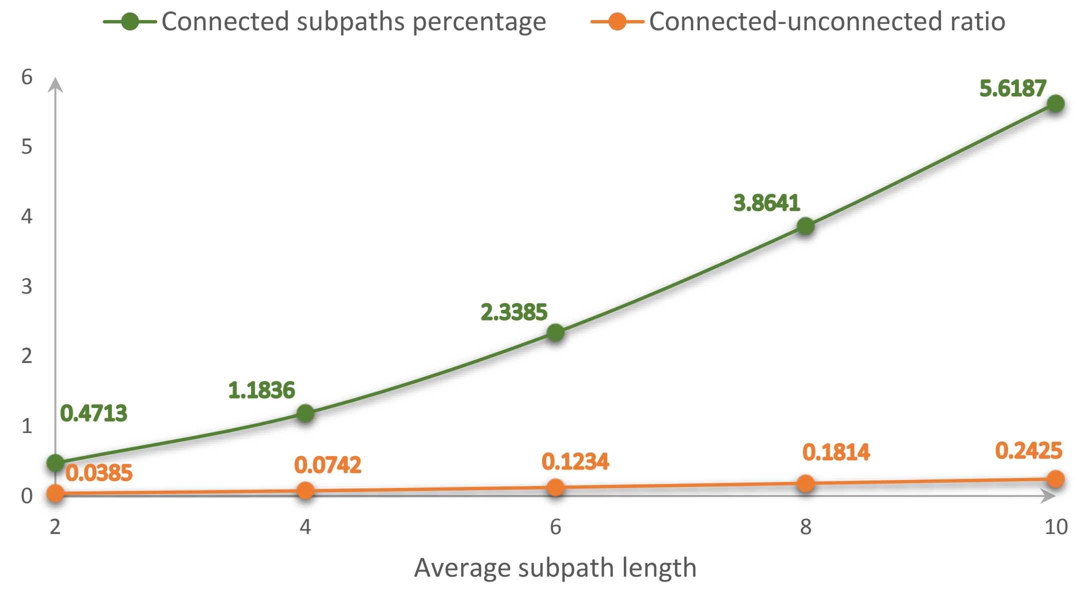


Figure 5.3: *The intra-subpath connectivity modelled as a function of the subpath length.*

5.2 Cornell box: light translation

This section examines another test performed on the Cornell box. Instead of translating the white cube, the current test translates the light source 2 units along the Z axis. The first frame of this animation sequence is rendered with bidirectional path tracing, while the other 9 frames are rendered with the path manipulation algorithm. Figure 5.4 depicts the 2^{nd}, 7^{th} and 9^{th} frame of the light source animation. The 512×512 frames were generated with 100 samples per pixel and 5000 samples per light source. The references were generated with $10{,}000$ samples per pixel via bidirectional path tracing.

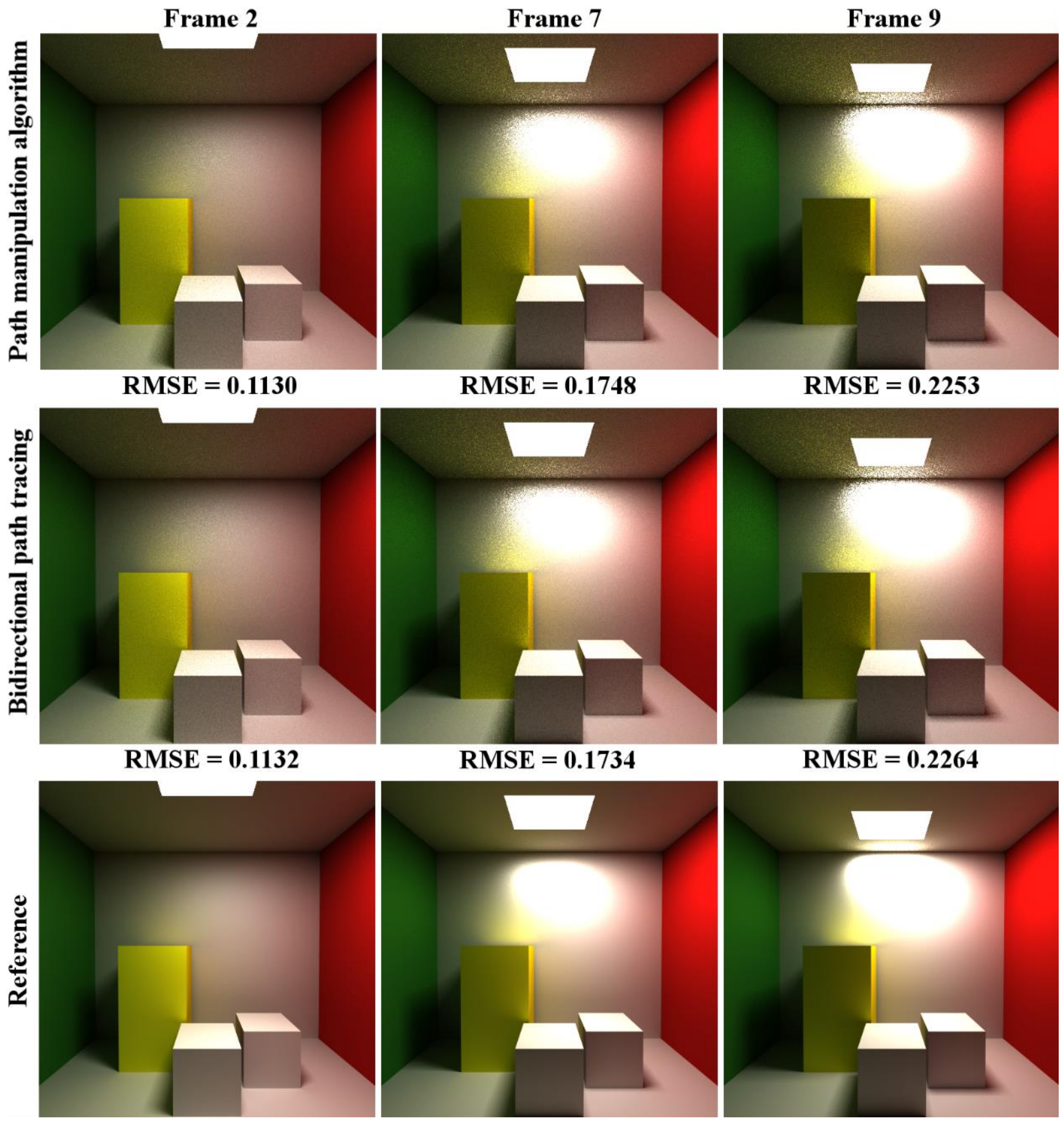


Figure 5.4: *Equal-sample comparison of the baseline and path manipulation algorithm.*

Algorithm / **Steps**	**Bidirectional path tracing**			**Path manipulation**		
	Frame 2	*Frame 7*	*Frame 9*	*Frame 2*	*Frame 7*	*Frame 9*
EPs generation	665.534s	710.402s	763.086s	—	—	—
LPs generation	116.113s	102.021s	102.357s	—	—	—
Invalid EPs reconstruction	—	—	—	66.773s	63.426s	66.752s
Invalid LPs reconstruction	—	—	—	0.126s	0.373s	0.417s
EPs regeneration	—	—	—	1.214s	2.202s	2.336s
LPs regeneration	—	—	—	0.112s	0.128s	0.11s
Collateral EPs identification	—	—	—	8.198s	4.287s	5.659s
EPs throughput evaluation	91.328s	37.763s	37.868s	2.995s	2.981s	3.807s
LPs throughput evaluation	0.014s	0.01s	0.011s	0.01s	0.011s	0.012s
Bidirectional contribution evaluation	1067.62s	1075.89s	1183.95s	1200.62s	1193s	1159.31s
Total execution time	**1940.76s**	**1926.24s**	**2087.43s**	**1280.79s**	**1267.22s**	**1239.23s**

Table 5.5: *Breakdown of the execution time employed by bidirectional path tracing, respectively by the path manipulation algorithm in generating three animation frames.*

5.2.1 Gain analysis

As discussed in chapter 3, section 3.6, the current test represents one of the worst-case scenarios for the path manipulation algorithm, since the entire buffer of eye subpaths must be re-evaluated. Table 5.5 reports the execution time for each step of the path manipulation algorithm. The obtained performance gain is prevalently the result of the path reconstruction phase. Though the invalid subpath identification routines trace the vast majority of rays, they are considerably faster than the generation of subpaths.

The rays need not be sampled, as they are determined by the existent subpath edges, and the anchors are computed only for a small number of invalid subpaths. The invalid subpaths include all the light subpaths, the out-of-scope and the in-scope eye subpaths.

The reduced number of invalid subpaths positively impacts the regeneration step, which requires between 1.33 and 2.45 seconds to execute. The collateral identification routines return all the eye subpaths for re-evaluation, as all of them are connected to the light subpaths generated from the single, altered light source. The reported time refers to clearing subpath fields, like the root radiance and the shadowing identifier (chapter 4, subsections 4.2.6 and 4.3.5). The average time gain of the reconstruction phase (Table 5.5, steps 3-7) over the generation phase (Table 5.5, steps 1-2) is 90.95%.

The throughput evaluation on the reconstructed subpaths maintains the performance gain and saves 92.90% of the execution time used to evaluate the throughput of all the subpaths. However, the contribution evaluation routine (chapter 4, subsection 4.2.5) processes the entire subpath collection and the execution time per frame is comparable to the one reported for bidirectional path tracing. The net gain of the path manipulation algorithm over bidirectional path tracing is 36.28% across the three animation frames.

5.2.2 Reconstruction analysis

Table 5.6 synthesizes the information gathered from the reconstruction of the 9 light animation frames. Note that the number of discarded rays is considerably smaller than the total number of generated rays. Hence, the in-scope identification routine traces the vast majority of rays. This fact is reflected by the execution time reported for the identification of invalid subpaths, which takes 99.09% from the total reconstruction time. The processed subpaths amount to only 7.24% from the total number of generated subpaths, of which 6.12% are $S0$ subpaths and 1.12% are unconnected subpaths. In this scenario, the $S0$ subpaths are obtained from the invalid eye subpaths whose primary rays directly intersect the light source, due to its sufficiently large surface. With an anchor on the light source, the $S0$ subpaths do not necessitate reconstruction and are thus less computationally expensive than the other invalid subpaths. Being the bulk of the processed subpaths, they justify the low reconnection time of 0.43 seconds.

Average	**#**	**%**
Invalid primary rays	655,308	0.5147
Invalid higher-order rays	1,243,384	0.9766
Discarded rays	991,046	0.7784
Total generated rays	127,310,450	100
Out-of-scope subpaths	1,562,511	5.9593
In-scope subpaths	336,172	1.2821
S0 subpaths	1,604,396	6.1191
Unconnected subpaths	294,295	1.1224
Processed subpaths	1,898,683	7.2415
Total generated subpaths	26,219,400	100
Subpath length	4	—
Invalid subpath identification time	62.7958s	99.0854
Reconnection time	0.4348s	0.6860
Total reconstruction time	63.3754s	100

Table 5.6: *The reconstruction information gathered from the 9 light animation frames.*

5.2.3 Bias analysis

Figure 5.5 illustrates the positive-negative (red-green) differences for the 2^{nd} and 7^{th} frame. The frames were generated with 100 samples per pixel and 5000 samples per light source. Each positive-negative difference is computed between the reference image and the frame generated either with the path manipulation algorithm or with bidirectional path tracing. Though the high-frequency noise is clearly visible, it does not differ significantly across the differently generated images. Like the root mean squared errors reported in Figure 5.4, the positive-negative differences indicate that the path manipulation algorithm exhibits a similar convergence behaviour to the bidirectional path tracing algorithm implemented in the light transport framework.

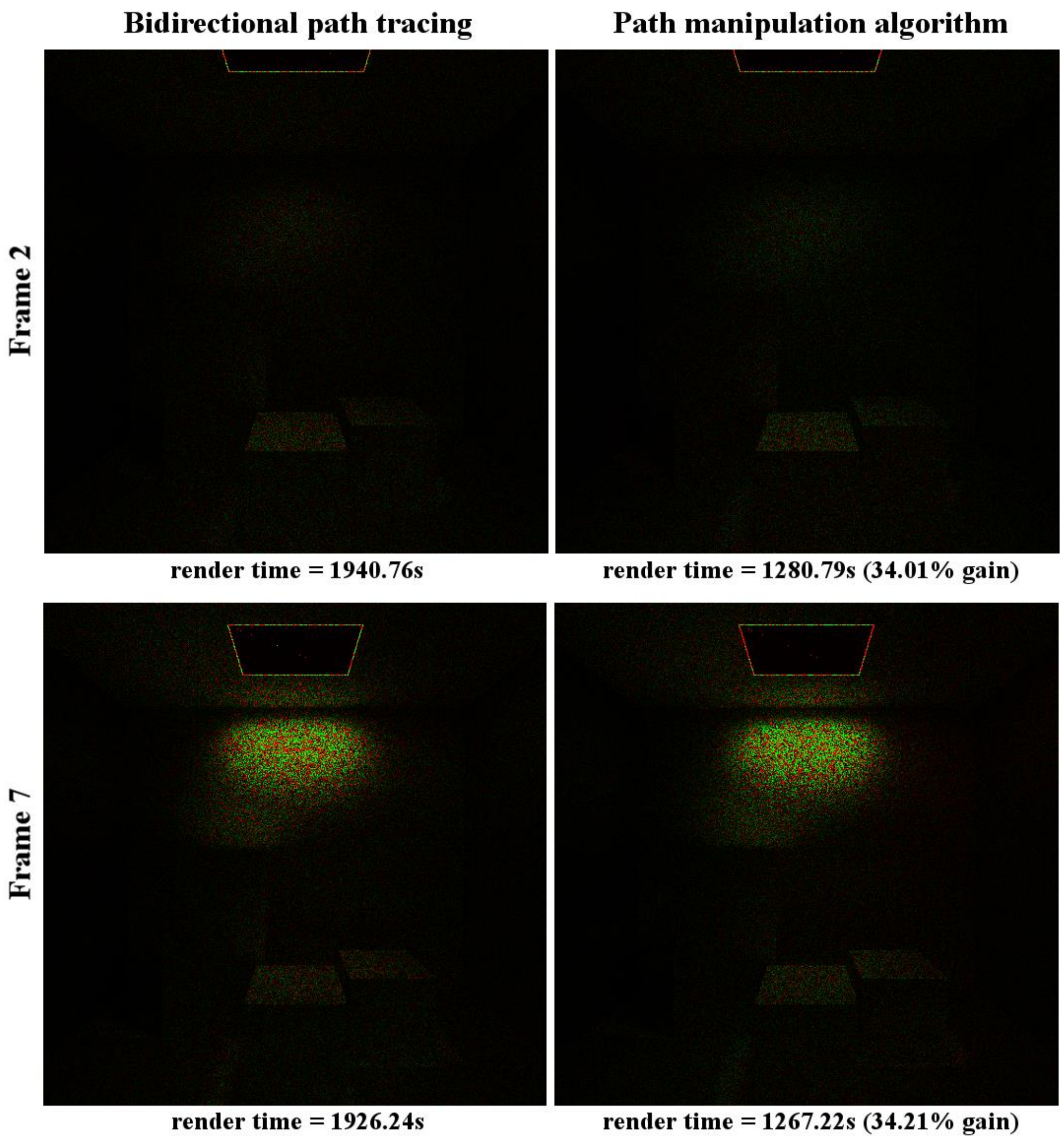

Bidirectional path tracing
Path manipulation algorithm
Frame 2
Frame 7
render time = 1940.76s
render time = 1280.79s (34.01% gain)
render time = 1926.24s
render time = 1267.22s (34.21% gain)

***Figure 5.5**: Comparison between the path manipulation algorithm and bidirectional path tracing using the positive-negative differences computed on two different frames.*

5.3 Cornell box: camera rotation

The last test executed on the Cornell box involves the rotation of the camera 10° around the Z axis. The camera animation subsumes eleven 512×512 frames, all rendered with the path manipulation algorithm, except the first frame which is rendered with bidirectional path tracing. Figure 5.6 depicts the 2^{nd}, 6^{th} and 11^{th} frame. Each frame was generated with 90 samples per pixel and 5000 samples per light source. The references were rendered with $10{,}000$ samples per pixel via bidirectional path tracing.

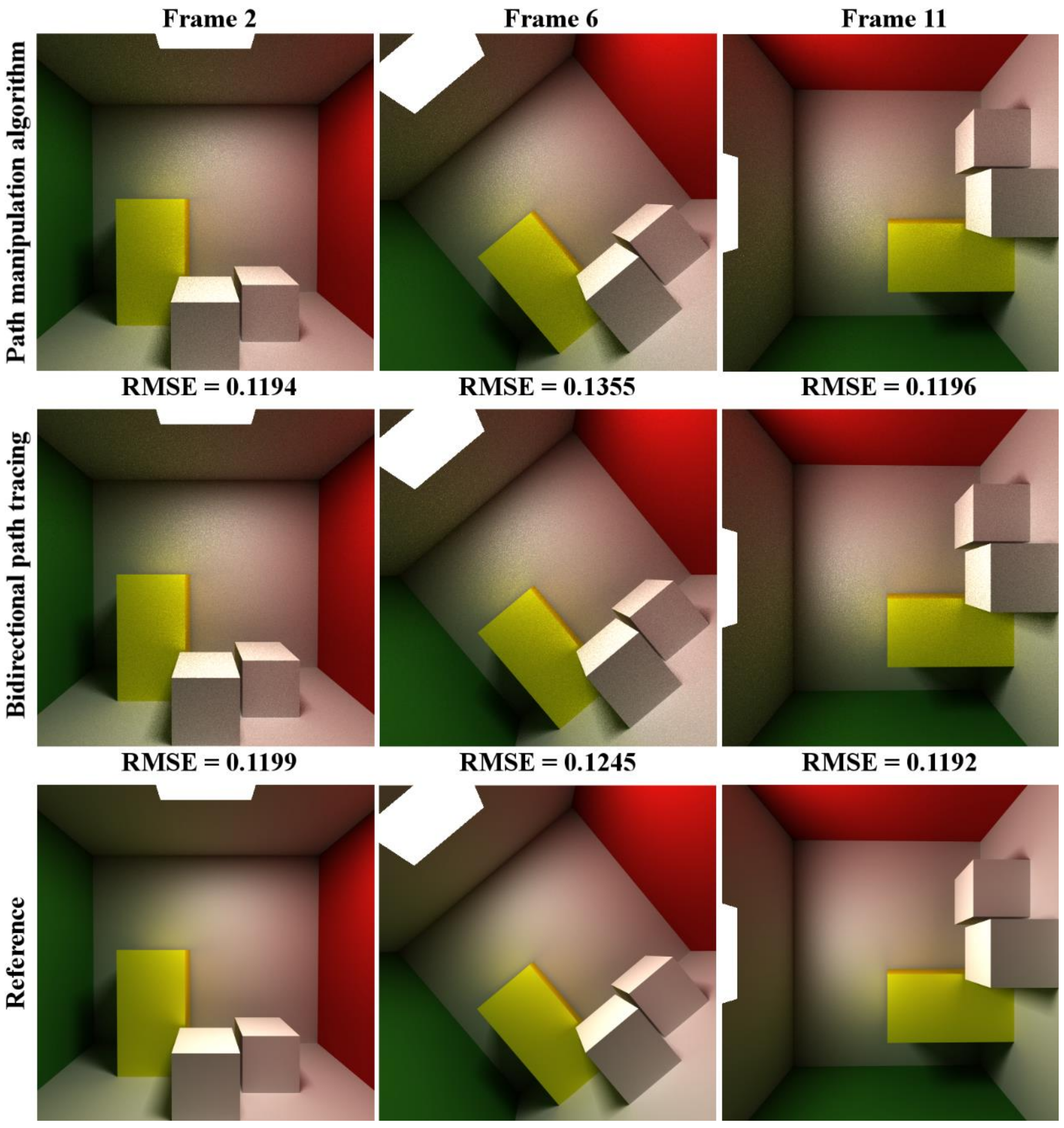


Figure 5.6*: Equal-sample comparison of the baseline and path manipulation algorithm.*

5.3.1 Gain analysis

The transformation of the camera represents another worst-case scenario for the path manipulation algorithm, because all the eye subpaths must be both reconstructed and re-evaluated. Table 5.7 compares the execution time consumed by each step of the path manipulation algorithm against the time engaged in executing each step of bidirectional path tracing. Analogous to the light animation, the performance gain obtained for the current test is preponderantly the result of the reconstruction phase. The light subpaths remain immutable and the time reported for their reconstruction is the time needed, by the in-scope identification routine, to check for disrupted edges.

Algorithm / **Steps**	**Bidirectional path tracing**			**Path manipulation**		
	Frame 2	*Frame 6*	*Frame 11*	*Frame 2*	*Frame 6*	*Frame 11*
EPs generation	598.446s	628.773s	693.547s	—	—	—
LPs generation	101.803s	101.961s	102.561s	—	—	—
Invalid EPs reconstruction	—	—	—	172.862s	179.846s	191.25s
Invalid LPs reconstruction	—	—	—	0.112s	0.148s	0.145s
EPs regeneration	—	—	—	147.083s	154.555s	163.575s
LPs regeneration	—	—	—	0.039s	0.009s	0.291s
Collateral EPs identification	—	—	—	0s	0s	0s
EPs throughput evaluation	35.378s	34.419s	35.462s	37.513s	39.889s	44.673s
LPs throughput evaluation	0.01s	0.01s	0.011s	0.001s	0.001s	0s
Bidirectional contribution evaluation	958.287s	982.067s	1053.56s	949.439s	942.079s	981.961s
Total execution time	**1694.08s**	**1747.39s**	**1885.3s**	**1309.59s**	**1319.43s**	**1384.83s**

Table 5.7*: Breakdown of the execution time employed by bidirectional path tracing, respectively by the path manipulation algorithm in generating three animation frames.*

Also, the regeneration of the light subpaths concerns mainly the subminimal subpaths. The collateral identification routines return immediately, since the collaterally invalid subpaths are excluded through the invalidation and reconstruction of all eye subpaths.

The reconstruction of the eye subpaths presents interesting aspects. The out-of-scope identification routine traces only the primary rays. The reconnection routine connects a small number of subpath chains and discards most of the higher-order rays. As such, the in-scope identification routine traces a reduced number of higher-order rays. Though the out-of-scope identification routine checks for higher-order dynamic vertices, it does not trace any higher-order ray, since it fails to detect such vertices.

Moreover, the reconnection routine is not executed for higher-order vertex levels, as no higher-order disrupted edge is identified. Hence, the reconstruction process traces all the primary rays, computes the primary anchors, reconnects a few of the latter to their secondary chains and examines the remaining higher-order rays for invalidations.

The regeneration routine invokes the path generation routine with most of the eye subpaths, yet the EPs regeneration time is $1/4$ of the EPs generation time per frame. The difference is that, for the regeneration step, the number of requested subpaths is already in place and the terminated subpaths no longer need to be replaced with new ones. During generation, the size of the active subpath buffer mostly equals the image resolution. Throughout generation, the size of the terminated subpath buffer increases proportionally with the requested number of samples per pixel. The regeneration routine processes the complete subpath buffer. A larger buffer of subpaths entails a larger buffer of rays, which ensures a higher rate of device occupancy and a better equipoise to the costs associated with the transfer of data over PCIe bus. As discussed in chapter 4 (subsections 4.2.1 - 4.2.2), all the rays are traced on the GPU. On the other hand, processing the complete subpath buffer obviates the replacement of terminated subpaths. The generation routine replaces with a new subpath any terminated subpath, whose affiliation count is below the specified number of samples. This operation involves the sampling of a new primary ray and the initialization of a new subpath. For the current test, suppressing the replacement of the terminated subpaths saves more than 23 million $(512 \times 512 \times 89)$ such operations. Overall the reconstruction phase gains 54.63% from the execution time used for the generation of all the subpaths.

The throughput evaluation for the eye subpaths exacts a comparable amount of time on both algorithms, yet the path manipulation algorithm circumvents this step for the valid light subpaths. The contribution evaluation routine processes all the generated subpaths and its execution time per frame is comparable across the two algorithms. The gain of the path manipulation algorithm over bidirectional path tracing is 24.58%.

Average	**#**	**%**
Invalid primary rays	23,592,960	20.6905
Invalid higher-order rays	3	0
Discarded rays	90,313,569	79.2032
Total generated rays	114,027,650	100
Out-of-scope subpaths	23,592,960	99.9788
In-scope subpaths	4	0
S0 subpaths	1,539,669	6.5245
Unconnected subpaths	22,053,294	93.4542
Processed subpaths	23,592,961	99.9788
Total generated subpaths	23,597,960	100
Subpath length	4	—
Invalid subpath identification time	90.8126s	50.6747
Reconnection time	87.3765s	48.7576
Total reconstruction time	179.2056s	100

Table 5.8: *The reconstruction information gathered from the camera animation frames.*

5.3.2 Reconstruction analysis

Table 5.8 reports the information gathered from the reconstruction of the 10 camera animation frames. Note that the number of invalid primary rays corresponds to the image resolution multiplied by the number of samples per pixel ($512 \times 512 \times 90$). The discarded rays constitute 79.20% from the total number of generated rays and most of them are eliminated during the single execution of the reconnection routine. As discussed, the reconnection routine is executed solely for the primary anchors. Hence, the indicated reconnection time quantifies the reconnection and deletion operations performed exclusively at the primary vertex level of the invalid subpaths.

The invalid subpath identification time quantifies several executions of the eponymous routines. The bulk of processing is performed throughout the first invocation of the out-of-scope identification routine, when all the regenerated primary rays are traced and the primary anchors are computed. The remaining invocations of the out-of-scope and in-scope identification routines, just search for dynamic vertices and disrupted edges. As most of the higher-order edges are discarded, 93.45% of the processed subpaths are unconnected and undergo regeneration. The figure reported for the $S0$ subpaths includes both the eye subpaths that intersect the light source and the eye subpaths that are reconnected from their primary anchors. However, the vast majority of the $S0$ subpaths regards eye subpaths whose primary rays directly intersect the light source, due to its sufficiently large surface. The $S0$ subpaths do not necessitate reconstruction and are less computationally expensive than the other invalid subpaths.

5.4 Computer-aided assembly design

The path manipulation algorithm reconstructs and reuses paths without being limited by the necessity of predefined animation paths. As long as the controller of the system (Reenskaug 1979) can determine from the user's actions the selected components and the intended transformations, the path manipulation algorithm can also work with input specified directly by the user. This characteristic is particularly important for applications that require direct interaction with the scene, like the simulation and design software used in commercial industries. Optis serves industries like aerospace, automotive industry, light design, architecture, energy design, beautification products, and consumer goods. These industries use their design phase simulations to drive the production process. Optis' physically-based renderer provides robust solutions at the cost of protracted executions (chapter 1, section 1.1). The main in-house challenge was to reduce execution time, while maintaining the simulation quality. Optis serves extensively the automotive sector, which relies on CAD/CAM systems to design its products. This test was devised for product assembling/disassembling sequences in CAD/CAM systems. The idea is to render physically plausible illumination effects in a dynamic, product design context, while reducing execution time. Better performance entails reduced production time and costs. The test consists of two 720×486 frames, rendered with 100 samples per pixel and 1000 samples for each of the 3 light sources.

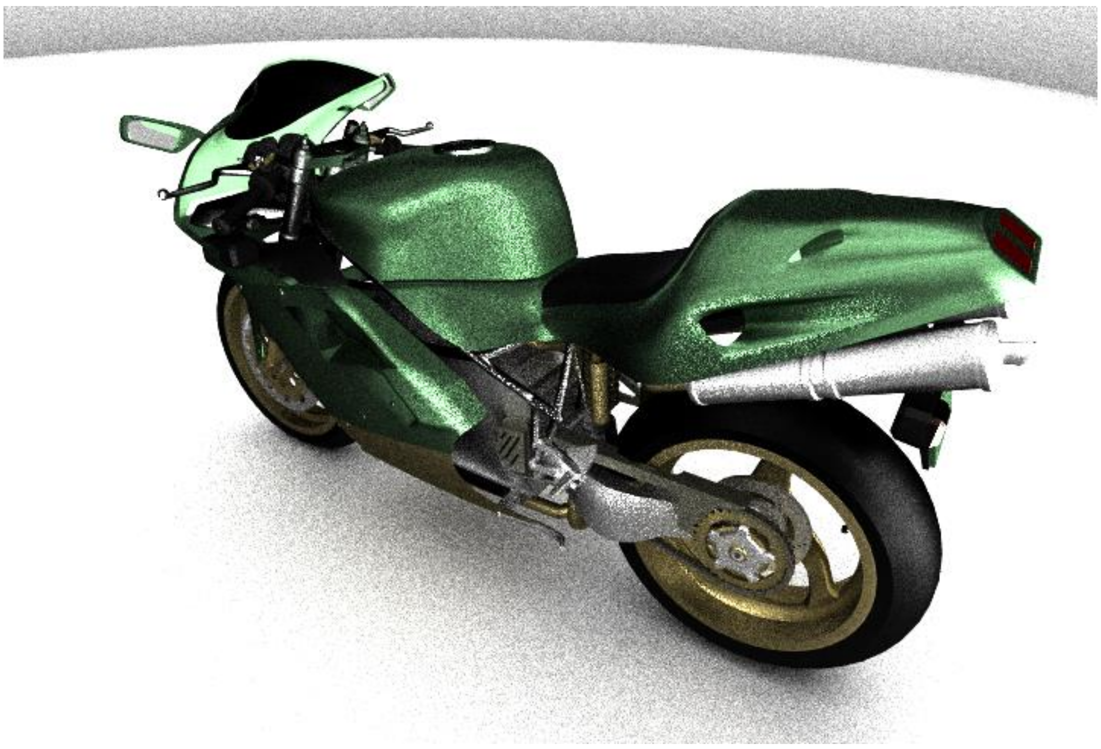


Figure 5.7: *The initial configuration from which product disassembling commences.*

5.4.1 Test configuration

The first frame contains the assembled product and represents the initial configuration. The second frame represents an instance in the disassembling process, in which various product components are transformed to expose previously hidden elements. The initial frame is rendered with bidirectional path tracing, whereas the second frame is reconstructed using the path manipulation algorithm. The transformations of the different components were defined to emulate the actions performed in assembly planning and they entail various translations and rotations. The current test lends itself to scalability analysis, as multiple parts are transformed and positioned to cover most of the image. The collateral subpaths were identified via the shadowing identifier and light subpath association routine. Figure 5.7 depicts the first frame of the product disassembling test, whereas Figure 5.8 compares the images of the second frame, rendered with both bidirectional path tracing and the path manipulation algorithm. The reference was generated with 40,100 samples per pixel via bidirectional path tracing.

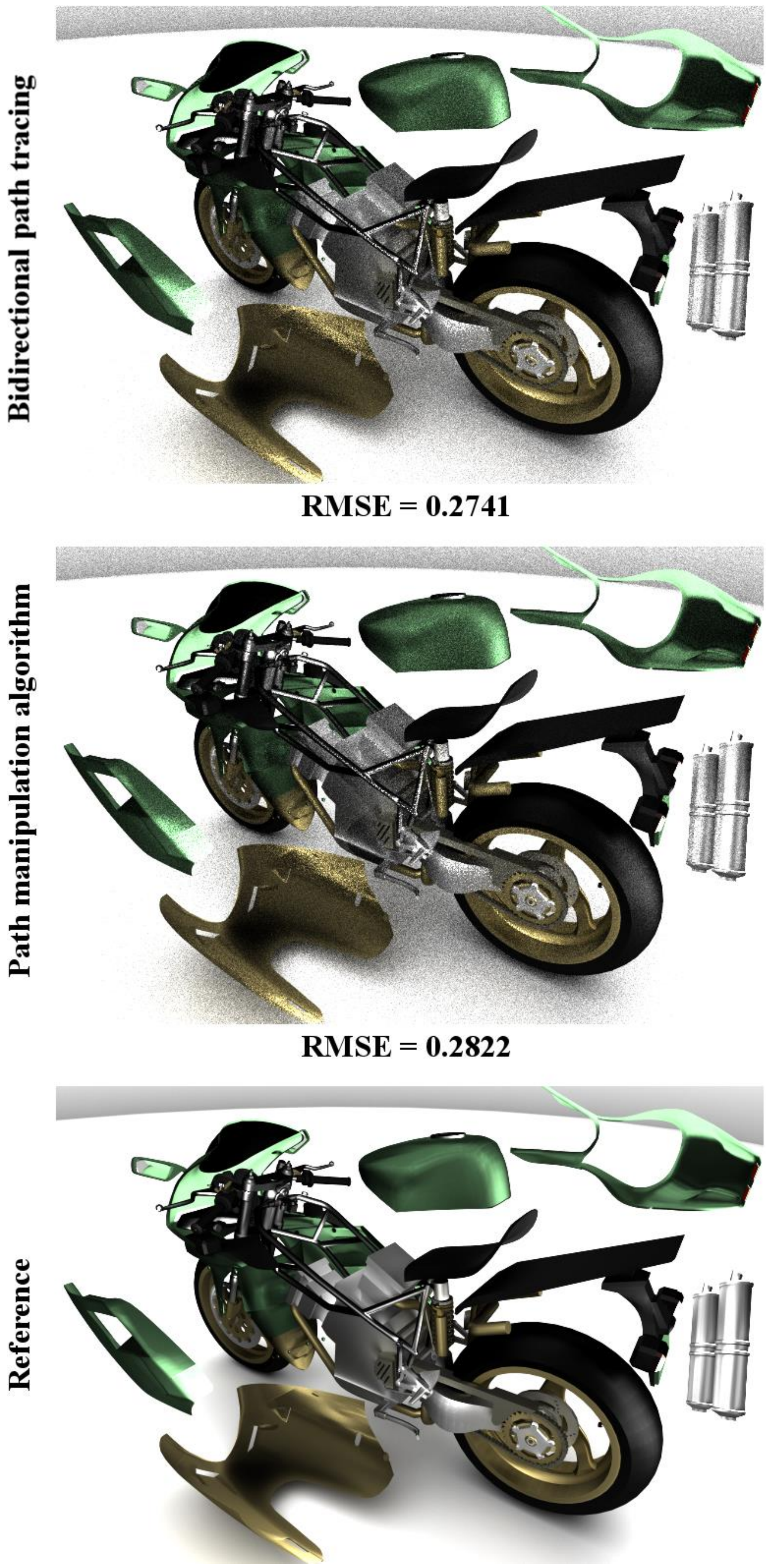


Figure 5.8*: Baseline and novel algorithm equal-sample comparison for the* 2nd *frame.*

Algorithm / Steps	Bidirectional path tracing	Path manipulation
	Frame 2	*Frame 2*
EPs generation	526.442s	—
LPs generation	182.137s	—
Invalid EPs reconstruction	—	134.69s
Invalid LPs reconstruction	—	1.559s
EPs regeneration	—	74.295s
LPs regeneration	—	0.326s
Collateral EPs identification	—	7.464s
EPs throughput evaluation	47.972s	16.541s
LPs throughput evaluation	0.032s	0.009s
Bidirectional contribution evaluation	493.711s	129.618s
Total execution time	**1250.52s**	**366.345s**

Table 5.9: *Breakdown of the execution time employed by bidirectional path tracing, respectively by the path manipulation algorithm in generating the disassembly frame.*

5.4.2 Gain analysis

To generate all the subpaths for the second frame, bidirectional path tracing employed 708.58 seconds. To reconstruct them, the path manipulation algorithm required only 218.33 seconds. Consequently, the time gain acquired by the path reconstruction phase (Table 5.9, steps 3-7) over the path generation phase (Table 5.9, steps 1-2) is 69.19%.

The evaluation of the throughput on the reconstructed subpaths (Table 5.9, steps 8-9) exhibits a similar performance gain and saves 65.52% of the time used to evaluate the throughput of all the subpaths. The contribution evaluation routine benefits from the reduced number of processed subpaths and extracts a performance gain of 73.75%. The reconstruction and evaluation phases gain 70.70% of the overall execution time. Figure 5.9 compares the two algorithms via the frame's positive-negative differences.

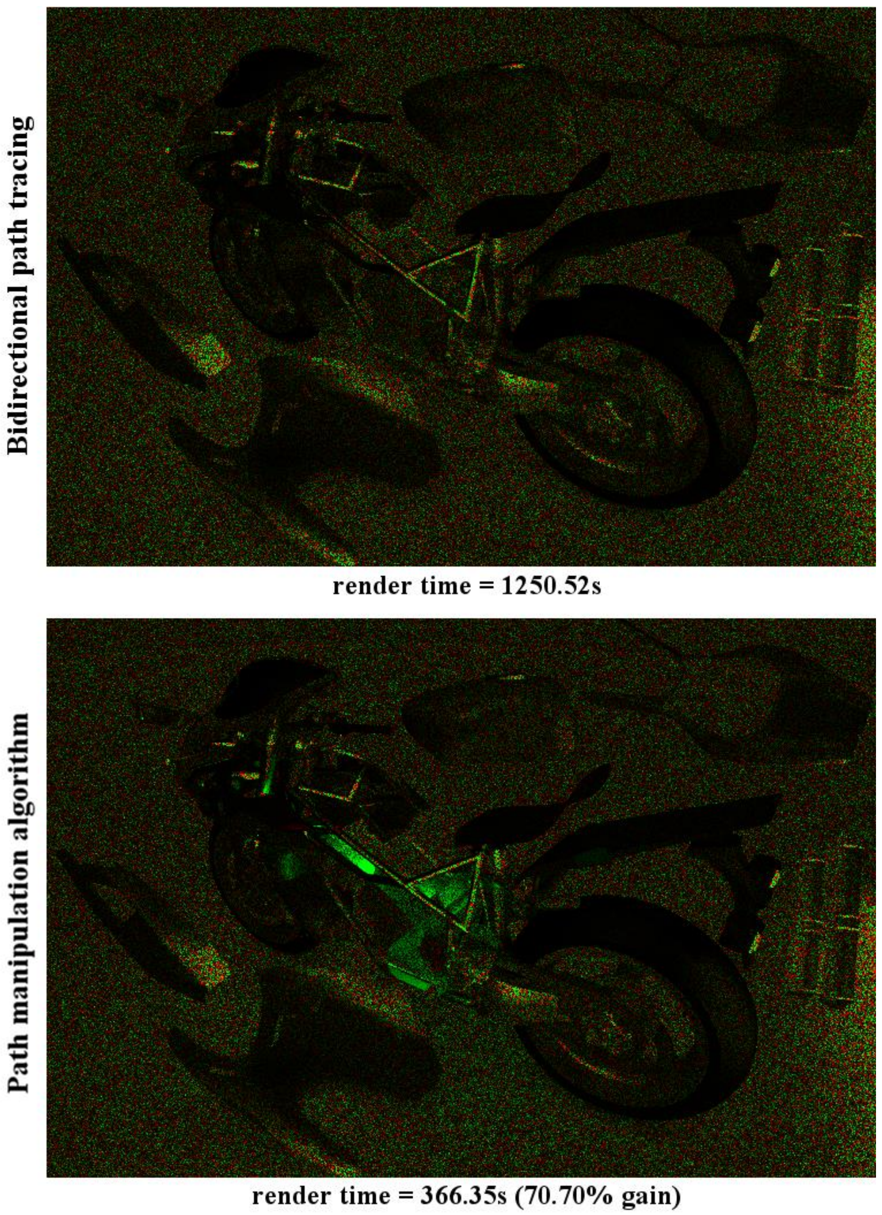

Bidirectional path tracing
render time = 1250.52s
Path manipulation algorithm
render time = 366.35s (70.70% gain)

***Figure 5.9**: Comparison between bidirectional path tracing and the path manipulation algorithm using the positive-negative differences computed on the disassembly frame.*

As mentioned in chapter 3, section 3.6, the path manipulation algorithm suffers from bias when the collateral subpaths are inappropriately reconstructed. In this case, the shadowing identifier and light subpath association routine does not suffice and certain collateral subpaths are not reconstructed. These unprocessed collateral subpaths cause bias (excessively green regions) by retaining the previous shadows. Note that the other image regions (bodywork, fuel tank, exhaust silencers, etc.) are correctly reconstructed. The bias could be completely removed by conventionally regenerating all the subpaths affiliated with the pixels of the affected regions, which would still be less expensive than the exhaustive regeneration of subpaths entailed by bidirectional path tracing.

Average	**#**	**%**
Invalid primary rays	11,804,075	12.9502
Invalid higher-order rays	2,041,465	2.2397
Discarded rays	21,199,686	23.2581
Total generated rays	91,149,533	100
Out-of-scope subpaths	6,749,255	19.2863
In-scope subpaths	7,095,050	20.2745
S0 subpaths	10,971,291	31.3510
Unconnected subpaths	2,871,670	8.2059
Processed subpaths	13,840,558	39.5501
Total generated subpaths	34,995,000	100
Subpath length	3	—
Invalid subpath identification time	92.1078s	67.6026
Reconnection time	42.3830s	31.1070
Total reconstruction time	136.2490s	100

Table 5.10: *The reconstruction information associated with the disassembly frame.*

5.4.3 Reconstruction analysis

Table 5.10 reports the information derived from the reconstruction of the second frame. The light sources used to illuminate the scene have surfaces wide enough to increase the probabilities of being randomly intersected by the eye subpaths. In fact, most of the connected subpaths are $S0$ subpaths. The reconnected subpaths amount to $7{,}207$. Like for bidirectional path tracing, the $S0$ subpaths constitute a performance source for the path manipulation algorithm, as they do not necessitate reconstruction and are less computationally expensive than the other invalid subpaths. The discarded rays, which in the second frame accrue to 23.26% from the total generated rays, are another performance source. The first frame was rendered in 1154.73 seconds, while the second frame was reconstructed in 366.35 seconds, yielding a net performance gain of 68.27%.

5.5 Interior architecture

The significance of Monte Carlo ray tracing increased with the engagement of the latter in a multitude of industrial sectors. Optis supports the design of energy-efficient lit buildings through accurate illumination and appearance simulations. Architecture uses extensively Monte Carlo ray tracing for accurate light transport simulations. In such a context, performance is another factor that directly impacts the production cycle. Fast executions imply additional flexibility in accommodating scene dynamism, an appealing trait in most sectors as it permits the transformation of the objects and an evolving perspective of the scene. The path manipulation algorithm supports scene dynamism by extending bidirectional path tracing to reuse paths in the temporal domain. The devised strategies reconstruct subpaths generically, regardless of whether the camera, a light source or another object was transformed. This test was ideated as an interior architecture scenario, in which different scene elements are altered to create a different perspective. By combining object, light source and camera transformations, the proposed algorithm is analysed in a realistic use case that emphasizes its features.

5.5.1 Test configuration

The test starts from an initial configuration and alters different scene objects across three frames. The initial frame is generated with bidirectional path tracing, whereas the other three frames are reconstructed with the path manipulation algorithm. The frames were rendered with a size of 640×512, using 80 samples per pixel and 200 samples for each of the 24 light sources. The references were generated with 10,240 samples per pixel via bidirectional path tracing. Figure 5.10 illustrates the first two frames generated for this test. The second frame represents the first step in the redesign of the illustrated building interior and it involves the repositioning of the mirror, together with all the elements that compose the washbasin. The perimeter method was used to identify the collaterally invalid subpaths. The insets displayed on the right column emphasize the changes determined by the repositioning of the objects. Both the cones of light cast on the tiled wall and the shadows cast by the washbasin are appropriately reconstructed. Figure 5.11 compares the images obtained by rendering this second frame with bidirectional path tracing and the path manipulation algorithm.

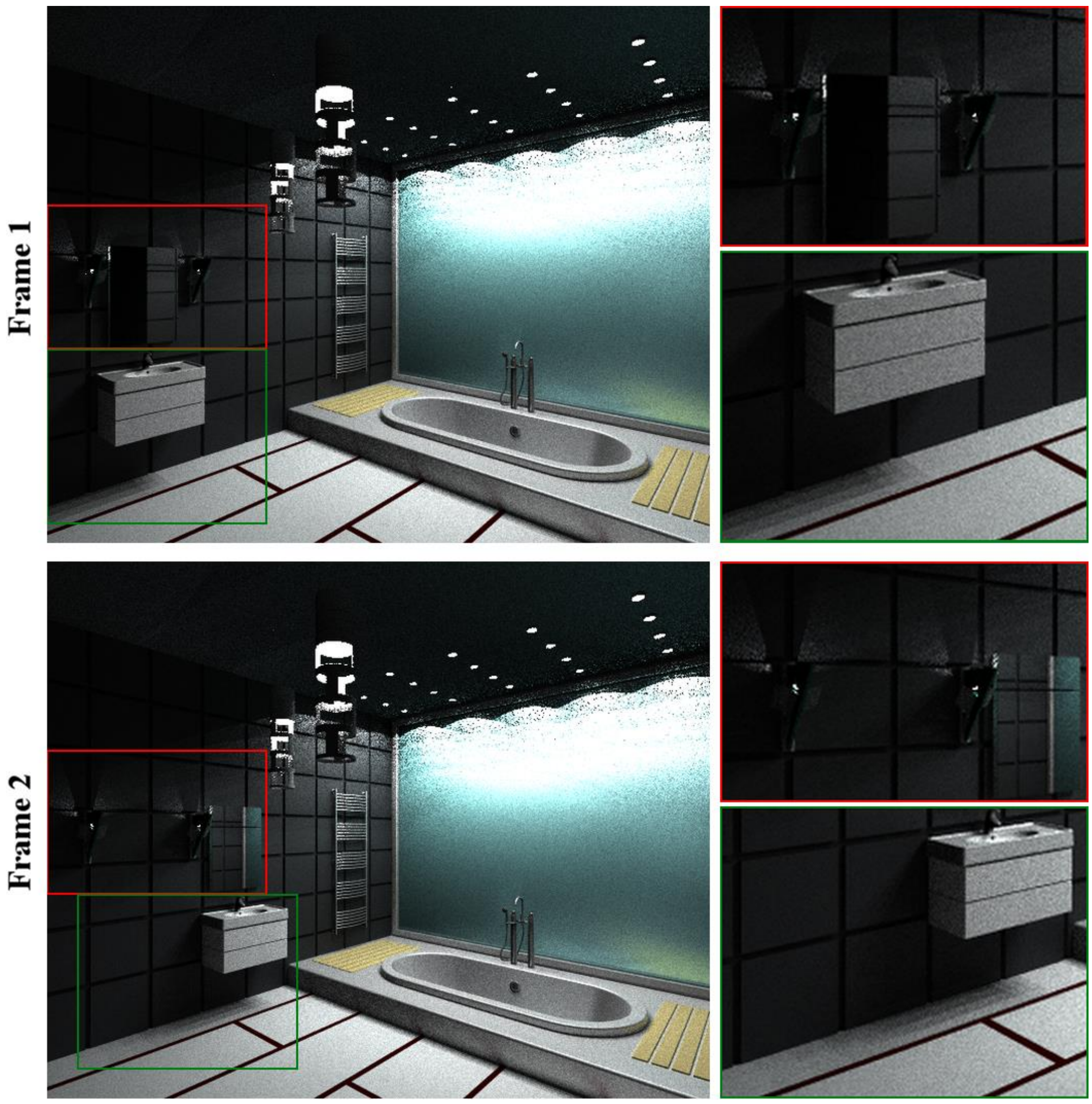


Figure 5.10: *The* 1^{st} *and* 2^{nd} *frame used in the adaptive redesign of a building interior.*

5.5.2 Frame 2: reconstruction analysis

The reconstruction process discarded 10.35% from the total of 124,347,106 generated rays. This percentage regards the number of eliminated rays, which are not traced by the identification routines. The percentages of invalid primary and higher-order rays are comparable and amount to 1.46%, respectively to 1.80% from the total number of generated rays. The total number of subpaths used to render each frame is 26,219,183. For the reconstruction of the second frame, approximately 15.45% of the total number of subpaths were processed. Out of these subpaths, 7.13% were identified as in-scope subpaths and 8.33% as out-of-scope subpaths. The unconnected subpaths equal 15.39% from the total number of subpaths, whereas the connected subpaths amount to 0.07%.

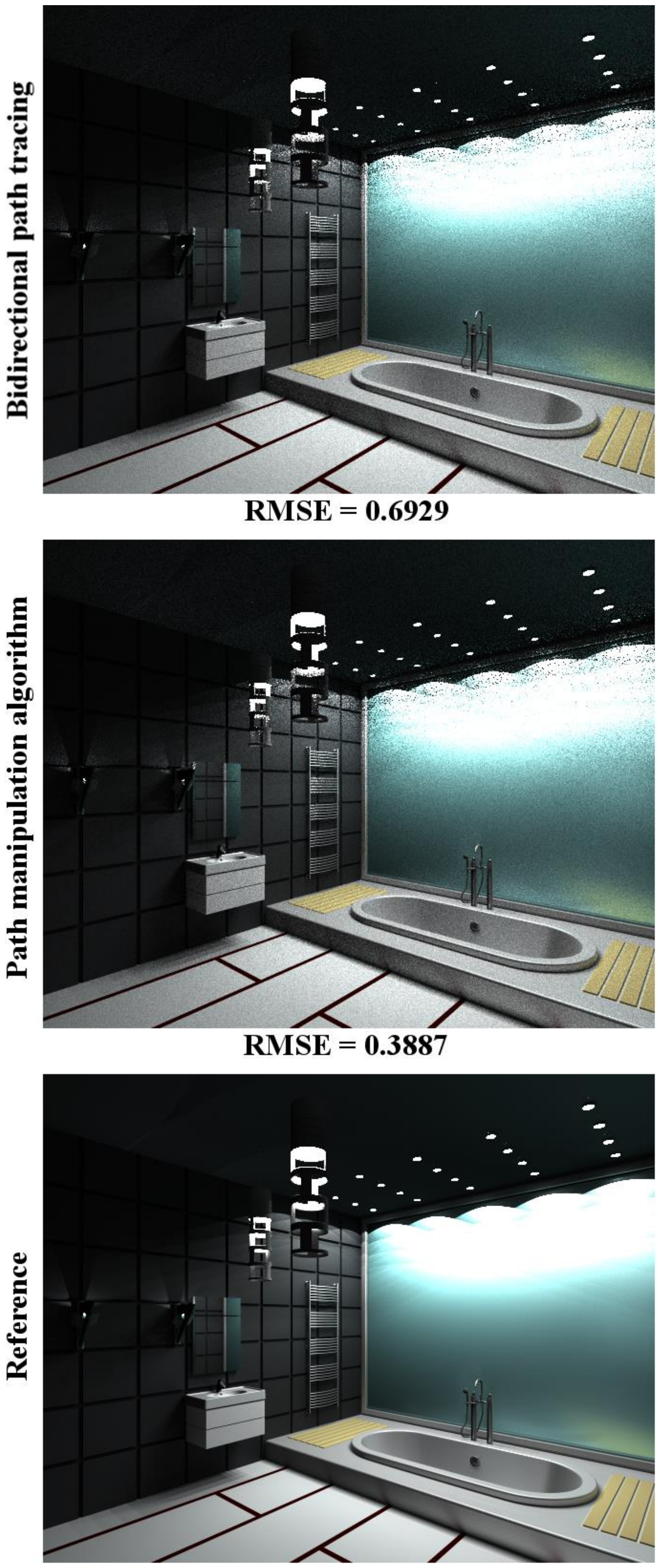


***Figure 5.11**: Baseline & novel algorithm equal-sample comparison for the 2^{nd} frame.*

The repositioning of the objects entails a wide translation (96 units on the X axis) and hence reduces the probabilities for subpath chains to be reconnected (subsection 5.1.3).

The initial frame required 5443.15 seconds to render, whereas the path manipulation algorithm reconstructed the 2^{nd} frame in 1157.58 seconds, gaining 78.73% execution time. The reduced number of processed subpaths positively impacts the regeneration step and the evaluation phase. The performance of the reconstruction process is further detailed in subsection 5.5.5. The average subpath length, for each rendered frame, is 4.

5.5.3 Frame 3: reconstruction analysis

The third frame of the current test represents the second step in the adaptive redesign of the bathroom and it involves the repositioning of the wall lamp geometry, with its associated light source, to the right of the mirror. For this frame, the collateral subpaths were identified via the shadowing identifier and light subpath association routine. Figure 5.12 displays the second and third frames, together with a zoom on some of the changes caused by the translation of the light source. Note the changes in the highlights and shadows cast by the altered light source. Figure 5.13 illustrates the third frame, generated with both bidirectional path tracing and the path manipulation algorithm.

The rays discarded throughout the reconstruction process amount to 1.10% from the total of 124,198,261 sampled rays. The invalid primary and higher-order rays equal 0.11%, respectively 0.32% from the total number of sampled rays. The number of subpaths used to generate the third frame is the same as for the other two frames. Out of the generated subpaths, 1.04% were identified as in-scope subpaths and 1.01% as out-of-scope subpaths, yielding a total of 2.05% processed subpaths. The connected subpaths equal 0.015% from the total number of subpaths, whereas the unconnected ones equal 2.032%. Like for the second frame, the translation step is wide and spans 190 units on the X axis. However, the connected-unconnected ratio improves due to the reduced size of the wall lamp. In fact, all the reported figures are low due to the size of the lamp. The third frame was reconstructed in 2171.66 seconds, which saves 60.10% of the time used to process the total number of subpaths for the first frame.

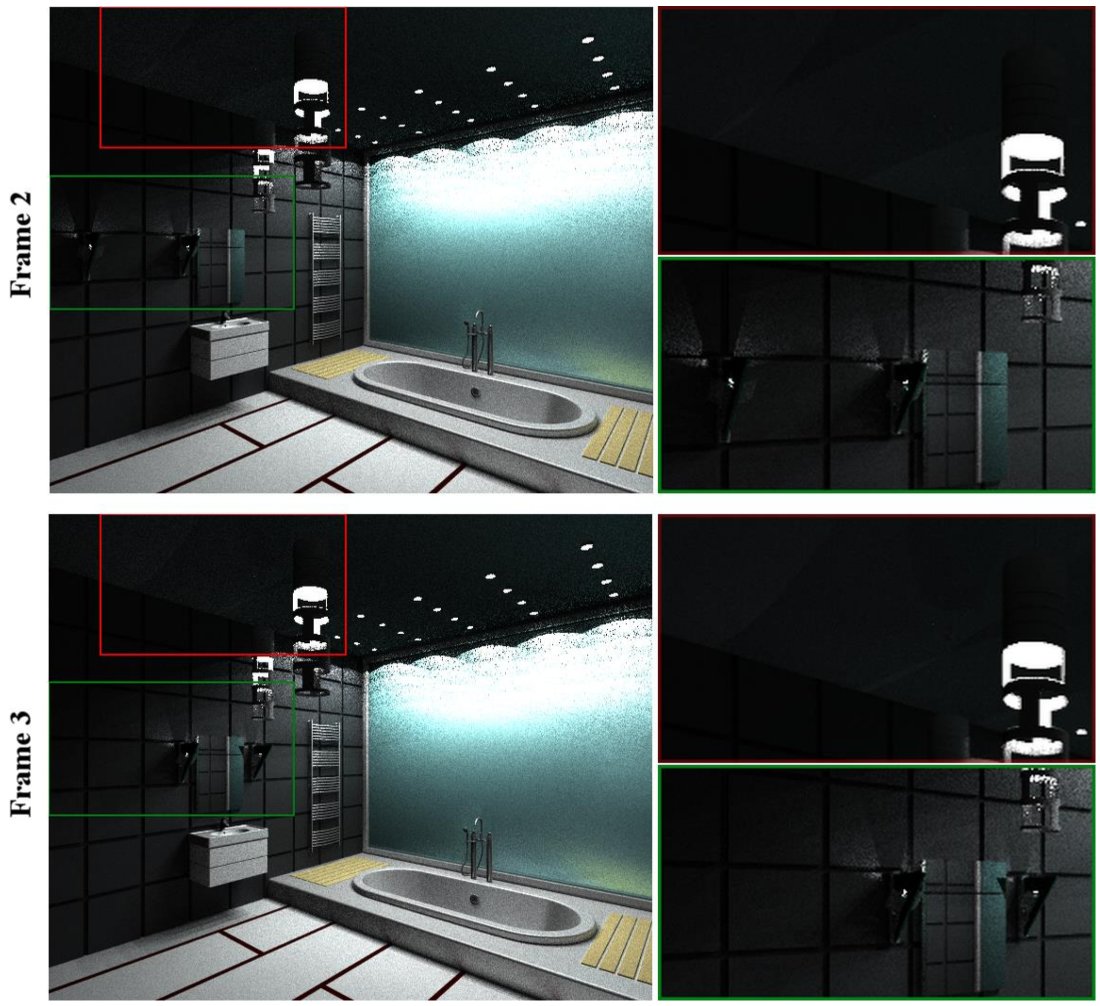


***Figure 5.12**: The results of the transformations applied in the* 2^{nd} *interior-redesign step.*

Compared to the second frame, the third frame loses in performance though 7.55 less subpaths are processed. The reconstruction process discards 11,507,756 less rays in the third frame, which is the number of extra rays tested by the identification routines. Also, more collateral subpaths are identified and thus the re-evaluation time increases.

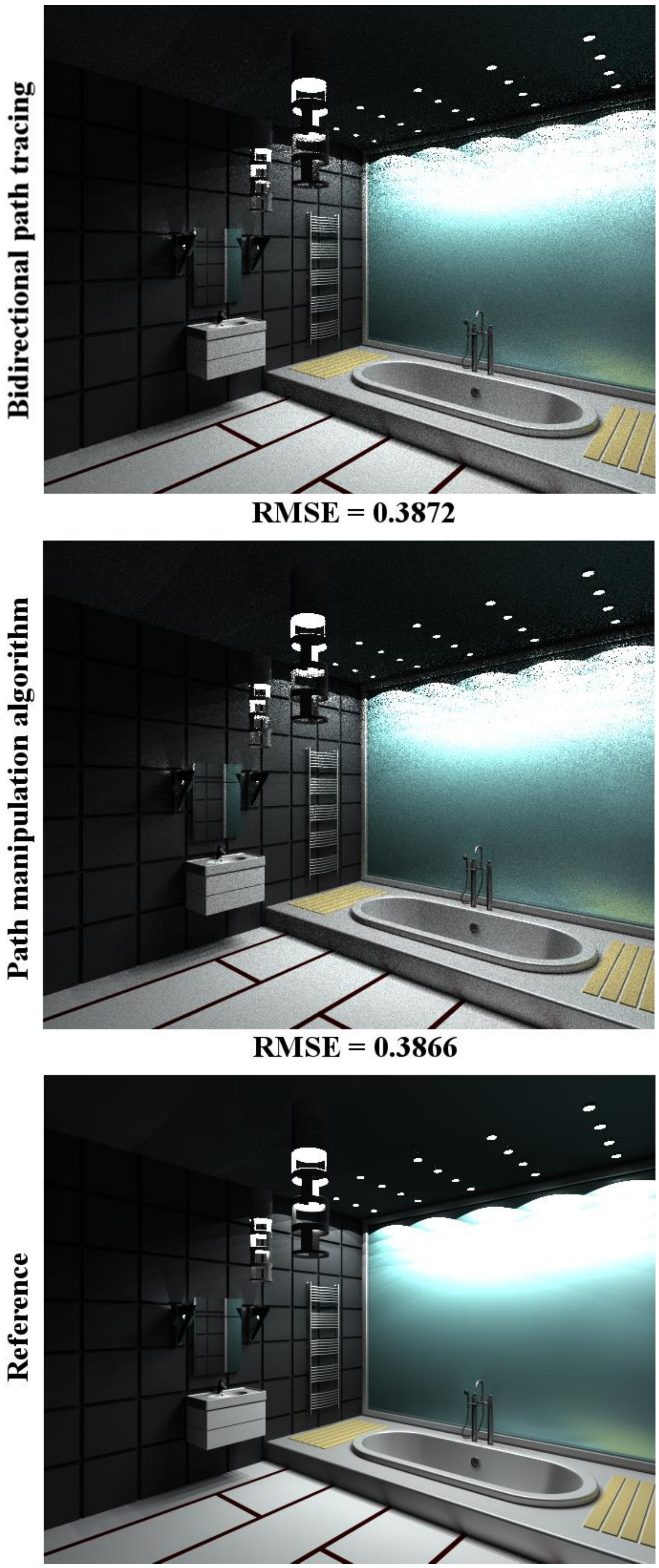


***Figure 5.13**: Baseline & novel algorithm equal-sample comparison for the* 3^{rd} *frame.*

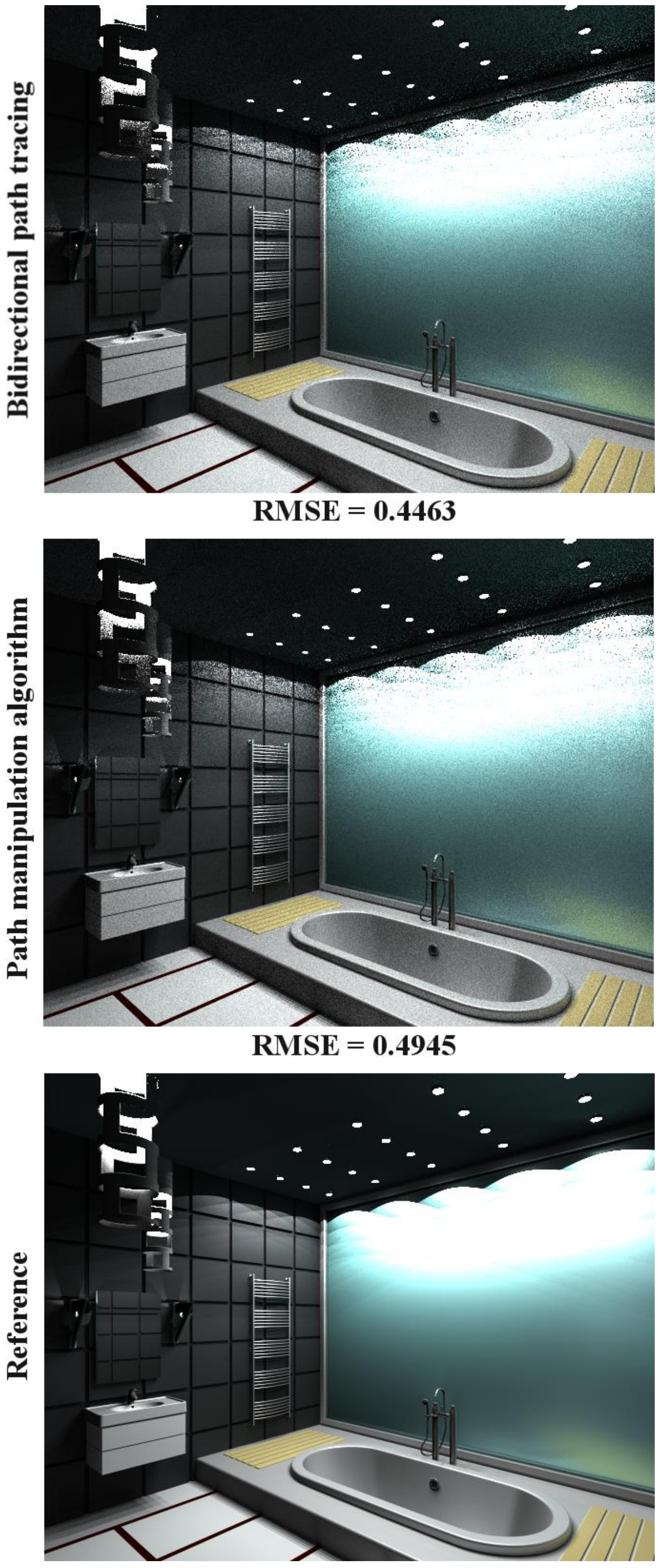


Figure 5.14*: Baseline & novel algorithm equal-sample comparison for the* 4^{th} *frame.*

5.5.4 Frame 4: reconstruction analysis

The last step of the redesign test consists in translating the camera 100 units closer to the main scene objects. Figure 5.14 displays the fourth and last frame, rendered with bidirectional path tracing and the path manipulation algorithm respectively. During the reconstruction of this frame, 78.88% of the rays were discarded, from the total of $124{,}254{,}126$ rays. The primary rays constitute the majority of invalid rays and they amount to 21.10% from the number of generated rays. Together the discarded and invalid rays accrue to 99.98%, which is exactly the percentage of processed subpaths. By moving the camera all the eye subpaths are invalidated, whereas the light subpaths remain unaltered and constitute the percentage of unprocessed subpaths. The total number of subpaths generated per frame is $26{,}219{,}183$. Out of these subpaths, 99.98% are out-of-scope, 0.47% are connected and 99.51% are unconnected. The fourth frame was reconstructed in 2565.92 seconds, gaining 52.86% from the time used to process the above number of subpaths. Compared to the 2^{nd} and 3^{rd} frames, the rendering of the 4^{th} frame discards considerably more rays and thus engages $1.72\times$, respectively $2.55\times$ less time in the reconstruction of invalid subpaths. Table 5.11 details the time employed to render each frame with the baseline and the path manipulation algorithm.

5.5.5 Gain analysis

The reduced number of processed subpaths positively impacts the reconstruction phase (Table 5.11, steps 3-7) of both the second and the third frame. The reconstruction of the 2^{nd} frame gains 44.36% execution time over the corresponding generation phase (Table 5.11, steps 1-2), whereas the reconstruction of the 3^{rd} frame gains 27.32%. Due to the extra number of rays traced by the identification routines, the reconstruction of invalid subpaths (Table 5.11, steps 3-4) takes $1.48\times$ longer for the 3^{rd} frame than it does for the 2^{nd} frame. Recall that 10.35% of the generated rays were discarded in the second frame, compared to only 1.10% discarded in the third frame. However, the larger size of the repositioned objects invalidates more subpaths in the 2^{nd} frame and triggers a longer regeneration step (Table 5.11, steps 5-6) as compared to the 3^{rd} frame.

Algorithm / **Steps**	**Bidirectional path tracing**			**Path manipulation**		
	Frame 2	*Frame 3*	*Frame 4*	*Frame 2*	*Frame 3*	*Frame 4*
EPs generation	1007.11s	1084.43s	1060.33s	—	—	—
LPs generation	238.854s	247.371s	232.989s	—	—	—
Invalid EPs reconstruction	—	—	—	595.789s	893.253s	337.837s
Invalid LPs reconstruction	—	—	—	20.733s	21.562s	21.389s
EPs regeneration	—	—	—	61.694s	22.136s	444.046s
LPs regeneration	—	—	—	3.868s	2.720s	2.836s
Collateral EPs identification	—	—	—	11.217s	28.245s	0.261s
EPs throughput evaluation	50.217s	50.216s	49.936s	17.940s	7.845s	76.265s
LPs throughput evaluation	0.309s	0.343s	0.32s	0.065s	0.061s	0.069s
Bidirectional contribution evaluation	4221.47s	3987.64s	4312.85s	445.006s	1195.21s	1677.93s
Total execution time	**5518.21s**	**5370.29s**	**5656.67s**	**1157.58s**	**2171.66s**	**2565.92s**

Table 5.11: *Breakdown of the execution time employed by bidirectional path tracing, respectively by the path manipulation algorithm in rendering the three redesign frames.*

From the reconstruction time reported for the 2^{nd} frame, 418.0 seconds were used to identify the invalid subpaths and 195.9 seconds were used to reconnect them. Likewise, 768.73 seconds and 142.37 seconds were used to identify and reconnect the invalid subpaths for the 3^{rd} frame. Compared to these two frames, the 4^{th} frame gains notable time in the reconstruction of the invalid subpaths. Specifically, 239.76 seconds were used to identify the invalid subpaths and 117.76 seconds were used to reconnect them. Yet, the fourth frame requires considerably more time in the regeneration step. The discarded rays (78.88%) account for this difference in execution time. Relative to the generation phase, the reconstruction phase of 4^{th} frame gains 37.65% execution time.

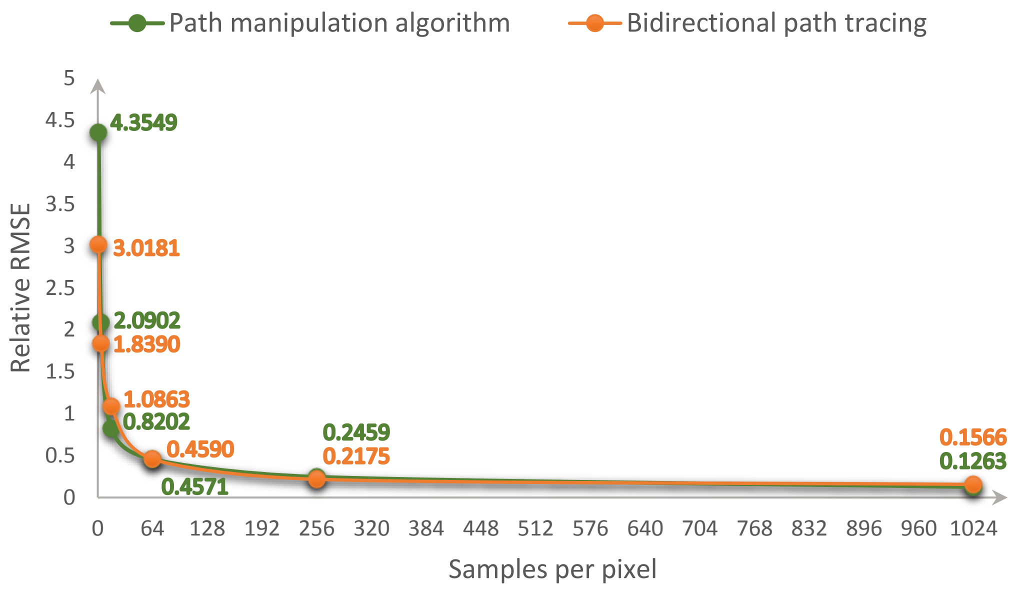


Figure 5.15: *The convergence behaviour shown by the path manipulation algorithm & bidirectional path tracing at rendering the* 3^{rd} *frame with increasing samples per pixel.*

The time reported for the throughput evaluation (Table 5.11, steps 8-9) is proportional to the number of processed subpaths and increases from the lowest value in the third frame to the highest value in the fourth frame. The collateral identification routine used in the third frame returns more subpaths for re-evaluation than the one used in the second frame, hence the longer execution time reported by 10^{th} step in Table 5.11. The net performance gain of the path manipulation algorithm over bidirectional path tracing is 79.02% for the second frame, 59.56% for the third frame and 54.64% for the fourth frame. Figure 5.15 illustrates the convergence behaviour of both the path manipulation algorithm and bidirectional path tracing, for the 3^{rd} frame of the interior redesign test.

5.6 Conclusions

This chapter investigated the performance and the statistical properties of the path manipulation algorithm. The first three sections examined the results obtained by transforming an object, the light source and the camera of the Cornell box. Subsection 5.1.3 presented the factors that influence the intra-subpath connectivity. Section 5.4 addressed the scalability of the algorithm by transforming various objects. Section 5.5 evinced its generality by combining object, light source and camera transformations.

In all tests, the path manipulation algorithm outperformed bidirectional path tracing and produced images with a comparable quality. The overall performance gain of the proposed algorithm over the baseline ranged from 24.58% to 79.02% across the five examined tests. The efficiency of the path manipulation algorithm is underlain by the lightweight invalid subpath identification routines (chapter 4, subsection 4.3.1) and by the intra-subpath connectivity strategy (chapter 4, subsection 4.3.2). The identification routines perform less operations than the path generation routine (chapter 4, subsection 4.2.3), whereas the reconnection routine reuses path information, obviates subpath regeneration and reduces the traceable edges. The number of subpaths invalidated and reconstructed by the proposed algorithm, also influences the overall performance gain.

The least efficient scenarios occur when the camera or the only light source of the scene is transformed. The transformation of the camera entails the reconstruction and re-evaluation of all the eye subpaths. Consequently, performance is drawn only from the reconstruction phase, since the evaluation phase performs the same operations for both the path manipulation algorithm and bidirectional path tracing. Another source of performance for the Cornell box camera rotation, besides the identification and reconnection routines, is the regeneration routine. As discussed in subsection 5.3.1, a better execution time is obtained by regenerating the entire subpath buffer and by suppressing the replacement of the terminated subpaths. A similar effect can be observed for the camera translation in the 4^{th} frame of the redesign test (Table 5.11).

The Cornell box contains one light source, whose translation causes the re-evaluation of all the eye subpaths. Due to the low number of reconstructed subpaths both the reconstruction phase and the throughput evaluation step exhibit high performance (subsection 5.2.1). Yet, the re-evaluation of all the eye subpaths reduces drastically the performance gain. The translation of only a small light source, from the 24 comprised ones, impacts less acutely the performance of the 3^{rd} redesign frame. Though the collateral identification routine returns a substantial number of subpaths, the re-evaluation of these subpaths is $3.34\times$ faster than the evaluation of all the eye subpaths (subsection 5.5.5). The reconstruction phase of the 3^{rd} redesign frame is notably less efficient than the Cornell box homologue. However, combined with the evaluation phase, it yields a better overall performance than the one of the Cornell box.

The best scenarios for the path manipulation algorithm are those that contain multiple light sources and transform other objects than the camera. Across the analysed tests, such scenarios (sections 5.1 & 5.4, subsections 5.5.2 & 5.5.3) exhibited a performance gain superior to 50%. Performance depends on the number of retraced edges, the percentage of connected/$S0$ subpaths, the number of regenerated subpaths, the subpath length, the number of re-evaluated subpaths and the scene configuration. The number of retraced edges determines the invalid subpaths identification time, whereas the connected subpaths reduce the regeneration necessity and increase path reuse. The subpath length affects both the regeneration and the re-evaluation process. The scene impacts performance through its light sources and materials. For example, scenes with large area light sources generate more $S0$ subpaths than the ones with small light sources. As they do not require reconstruction, $S0$ subpaths are a performance source. Similarly, the scattered energy and materials influence the intra-subpath connectivity.

The inspected tests are emblematic of the tests executed during the current research. For instance, the disassembly test uses similar concepts to a test carried on Newton's cradle, which applies combined affine transformations to the suspended spheres. The 3^{rd} redesign frame emulates another test, in which a desk lamp, with its disk light source, is translated and rotated to illuminate various desk regions. The Cornell box was used to translate two cubes instead of one. Regardless of the light sources and materials, all the tests exhibited increased performance and comparable image quality.

The path manipulation algorithm can benefit various industrial sectors and preliminary discussions were carried with Optis, to integrate it in the in-house rendering software.

Chapter 6

Future developments

The path manipulation algorithm extends bidirectional path tracing to reuse paths in the temporal domain, with the purpose of generating a variety of illumination effects in conditions of dynamic geometry. Its core is an intra-subpath connectivity strategy, which reconnects the disrupted chains of an invalid subpath based on the scattering and stochastic properties of the connecting vertices. A tentative connection with insufficient throughput, along the edge that connects the subpath chains, is rejected without introducing bias. Also, an intra-subpath connection is established only if the probabilities for scattering to occur, in and from the direction of the connecting edge, pass the rejection test. A failed intra-subpath connection causes the subpath to be regenerated from its unconnected anchor. Chapter 5 analysed the statistical properties and the performance gain of the novel algorithm. For the same number of samples, the execution time gain of the path manipulation algorithm ranged from 24.58% to 79.02%. This chapter reflects on the intra-subpath connectivity strategy, proposes performance improvements, addresses current limitations and suggests future avenues of research.

Currently, the path manipulation algorithm discards the vertices to which an anchor failed to be connected. The vertices that succeed an anchor compose the secondary subpath chain and the disposal of more than two vertices results in eliminating such a chain. The connectivity percentage could be improved by storing these secondary chains and using them to establish intra-subpath connections with other unconnected anchors. The simplest approach would be to randomly assign a secondary chain to an unconnected anchor and attempt to reconnect them via an intra-subpath connection. Yet, this approach may not considerably increase the connectivity percentage due to visibility issues, insufficient throughput or low intra-subpath connection probabilities.

Another approach would be to extrapolate some of the concepts that underlie the probabilistic connections (Popov et al. 2015) and compute a probability mass function based on the contribution of the tentative intra-subpath connection. The probability mass function, for a given subpath, could be computed by normalizing the tentative connection contribution with the sum of the contributions of all the connections that could be established between the unconnected anchor and the stored secondary chains. The inverse transform sampling method could be used to select the actual secondary chain. Once connected to an anchor, a secondary chain would be removed from the pool of chains. Yet, this approach may be expensive if there are many secondary chains and unconnected anchors. An option would be to sample a smaller number of secondary chains from the initial pool and use only those chains to compute the probability mass functions for a given anchor. The throughput and probability density for a reconnected subpath would have to consider the probability mass function rather than probability (3.32). By computing the probability mass function for each unconnected anchor, the introduced correlation would have less effect on variance than the interpolation of the probability mass functions from proximate anchors. In fact, correlation would regard the efficient extraction of additional samples (Veach 1998, p. 307). Hence, the paths constructed from the reconnected subpaths could be evaluated as detailed in chapter 3, subsection 3.1.2. Variations of these approaches may also be implemented and used as concrete bases to determine the trade-offs between performance and code complexity.

The performance of the current system could be further improved by developing a space-partitioning data structure that would correlate the scene geometry with the path information. Currently, the in-scope routine identifies the invalid subpaths by tracing all of the generated rays, except for the discarded ones. This computational effort could be reduced by inserting, in the cells of an acceleration structure, both geometric and path information. For instance, a cell could associate each primitive that contains a subpath vertex with the corresponding subpath identifier and vertex level. The i^{th} vertex level regards the i^{th} vertex of a subpath. Methods in fast ray tracing (chapter 2, subsection 2.6.2) either update/rebuild the dynamic parts of an acceleration structure or rebuild only the acceleration structures composed entirely of dynamic objects. A cell selected for update/rebuild would then trigger the rebuild of the list containing the subpath identifier-vertex level pairs and the retracing of the rays associated with the indicated vertices. Such an approach would eliminate the exhaustive ray retracing and potentially the out-of-scope identification routine. A subpath vertex has incident and exitant rays and their retracing may identify both in-scope and out-of-scope subpaths.

Storing the light and eye subpaths in the main memory can become infeasible for higher image resolutions and numerous subpaths per pixel or light source. The tests examined in the previous chapter used between 80 and 100 subpaths per pixel to render images with resolutions from 0.25 to 0.33 megapixel. These configurations occupied 80% to 99% of the physical memory. Figure 6.1 illustrates the near-linear increase in the usage of the physical memory, that was caused by rendering the three scenes used in the previous chapter, with an increasing number of subpaths per pixel.

The memory limitations could be alleviated by implementing a caching scheme that stores in memory some of the subpaths and saves the rest on the disc. For example, the image could be divided in blocks of pixels and the requested number of subpaths could be simultaneously generated for all the pixels in a block. The pixels for which the number of subpaths reached the specified threshold, could be replaced with other pixels. The affiliation (chapter 4, subsection 4.2.3) would keep track of the generated subpaths per pixel, would trigger the replacement of the terminated subpaths with new ones and would determine the replacement of the pixels whose subpaths are all terminated. The terminated subpaths could be stored on and retrieved from the disc using threads. Like the subpath generation, the throughput and contribution evaluation could be executed on subpath blocks. The current implementation already evaluates the throughput and the path contributions, by diving the subpath buffer in blocks, based on the number of subpaths per affiliation. The caching scheme would require the redesign of the throughput and contribution evaluation steps, so that a subpath block could undergo both steps before being stored on the disc. The same processing and caching schemes could be used with the light subpaths, by grouping light sources instead of pixels. As evinced in chapter 4, section 4.3, the path manipulation algorithm replaces only the path generation phase of the standard Monte Carlo pipeline, leaving the other phases unaltered. Subpaths could also be reconstructed by diving, storing and retrieving the subpath buffer in blocks. The size of a block could be set to the one used for the generation of subpaths, or it could be assigned multiples of the warp size.

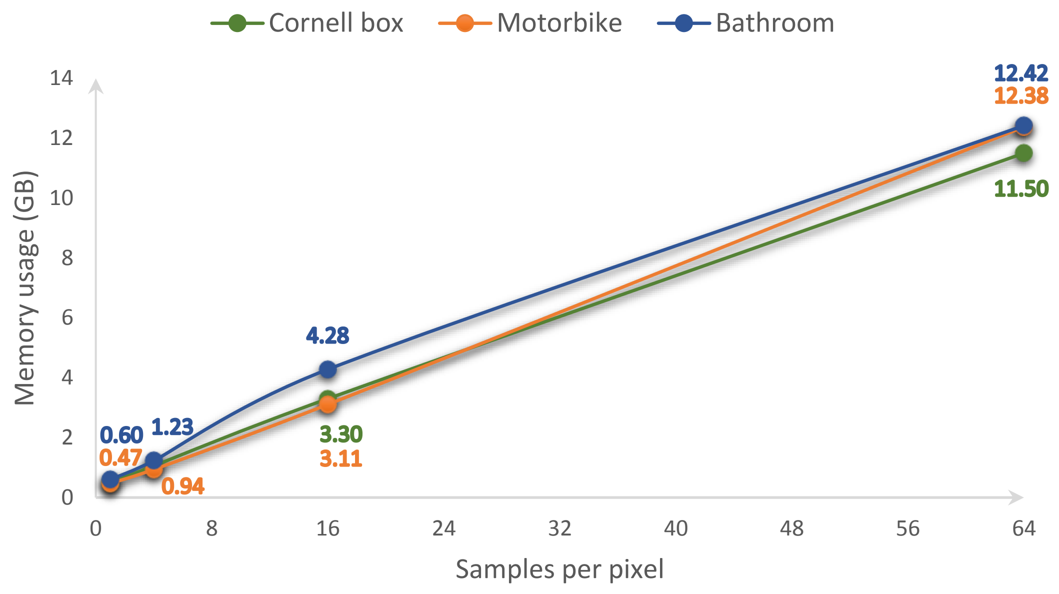


***Figure 6.1**: The total memory usage associated with rendering 1-64 subpaths per pixel.*

Currently, subpaths are reconstructed by dividing the subpath buffer based on the number of subpaths per affiliation. The remaining steps would be to store the blocks and retrieve all the reconstructed subpaths, so that they can be processed by the next phases. The blocks could be stored and retrieved asynchronously, as other blocks would be processed, with the advantage that I/O latencies would be minimized. The invalid subpaths could be stored, in a separate file, as they would be identified and reconstructed. This approach would facilitate the retrieval of blocks composed entirely of reconstructed subpaths and thus the processing of the latter by the subsequent phases of the path manipulation algorithm. The identification of the most appropriate storing/retrieval approach would require further investigation (Eisenacher et al. 2013). The identification of collateral subpaths would also benefit from further investigation that would stabilise it and would preclude scenarios affected by bias such as test 5.4.

The frames generated for the Cornell box tests were collated in AVI files and were inspected for artefacts caused by the loss of temporal coherence, such as flickering. The videos did not exhibit such artefacts. Producing animated sequences for more complex environments, analysing their temporal coherence and potentially combating temporal artefacts constitute another line of investigation. Lastly, it would be interesting to integrate the intra-subpath connectivity strategy with other point sampling methods.

Chapter 7

Conclusions

The current work introduced the path manipulation algorithm as a tool that extends bidirectional path tracing to reuse paths in the temporal domain. The previous chapters demonstrated that the path manipulation algorithm effectively addresses the restriction of static geometry commonly encountered in Monte Carlo light transport simulations.

Bidirectional path tracing can generate a variety of illumination effects for an extended range of complex lighting, scattering and geometric models. Its robustness stems from optimally combining paths, generated with different local path sampling techniques, in low-variance estimators. The local path sampling techniques together with multiple importance sampling are the key mechanisms that underlie bidirectional path tracing.

Though versatile, bidirectional path tracing is computationally expensive. As reported in chapter 5, the sampling operations and contribution evaluations required orders of minutes to produce results with moderate levels of variance. Since it does not support dynamism, bidirectional path tracing discards the generated paths immediately after the evaluation of their contributions and recomputes the illumination solution for the slightest scene transformation. Hence, each frame is rendered with the same timescale.

The balance between the accuracy and the time complexity of the Monte Carlo light transport simulation bears considerable significance for numerous industries. Chapter 5 demonstrated that the path manipulation algorithm outperforms bidirectional path tracing. On the one hand, its performance gain can be used to generate additional paths and thus further reduce variance. On the other hand, the path manipulation algorithm provides more flexibility by supporting dynamism and an evolving scene perspective.

Most path reuse algorithms (chapter 2, subsection 2.6.3) support dynamism by relying on limiting assumptions, which range from restricting the type of dynamic object to requiring predefined animation paths. Chapter 3 showed that the path manipulation algorithm reconstructs and reuses paths regardless of the subpath or scene dynamism type and without the need for predefined animation paths. Light and eye subpaths are processed independently of the camera, light source or other object transformations. This flexibility is underlain by the path manipulation strategies, which reconstruct subpaths generically and extend their lifespan to a generation-evaluation-reuse cycle.

The concepts of path validity and immutable contribution were introduced to identify the subpaths that can be immediately reused in the rendering of the transformed scene. The subpaths which breach the validity criteria are identified as invalid and the first step in their reconstruction is to compute their anchors. The location of the anchor on an invalid subpath was used to describe three reconstruction scenarios. The primary and terminus anchor reconstructions process the antipodal cases when the anchor either follows the first vertex or replaces the last vertex of an invalid subpath. The two-chain reconstruction processes subpaths split into two chains around the anchor.

Chapter 3 evinced that the path manipulation algorithm obviates the regeneration of the entire path collection, ports bidirectional path tracing to the temporal domain and supports dynamism. The core of the path manipulation algorithm is an intra-subpath connectivity strategy, which reconnects two disrupted chains of the same type into a functional subpath. The novel strategy uses the scattering and the stochastic properties of the connecting vertices to determine whether an intra-subpath connection can be established on the invalid subpath. Tentative connections with insufficient throughput are discarded without introducing bias. Similarly, the stochastic viability of a tentative connection is determined via a two-dimensional rejection test. Unlike conventional path generation methods, which use Russian roulette only to control the subpath lengths, this strategy uses such a test also during the connection process. A successful tentative connection generates a subpath in more than one piece by reusing existing information. As a path generation strategy, the intra-subpath connection does not affect the computation nor the weighting of the path contributions. A subpath which fails to meet the intra-subpath connection criteria, is regenerated from its unconnected anchor.

Bidirectional path tracing and the path manipulation algorithm are comprised in the light transport framework (chapter 4). The latter provided Optis with the first-ever implementation of bidirectional path tracing and of two state-of-the-art methods. Subpaths are generated using a similar approach to Novák et al. (2010), except that they are terminated via the Russian roulette test proposed by Veach (1998, p. 309) and the primary rays of the restarted subpaths are always resampled. Multiple paths are efficiently generated from the same pair of subpaths (Veach 1998, p. 300), with the advantages of zero costs and reduced variance. However, each eye subpath vertex is connected to a random light subpath (Novák et al. 2010), to reduce the deterministic noise and path correlation. The variance-reduction weights, are also computed via the recursive multiple importance sampling schema proposed by van Antwerpen (2011b).

These state-of-the-art techniques improve both accuracy and performance. Still, for a comparable image quality the path manipulation algorithm outperforms bidirectional path tracing, while supporting dynamism. Chapter 5 showed that the efficiency of the novel algorithm is underlain by the lightweight invalid subpath identification routines and by the intra-subpath connectivity strategy. The identification routines execute fewer operations than the path generation routine, whereas the reconnection routine reuses path information, avoids subpath regeneration and reduces the traceable edges.

The path manipulation algorithm can benefit various industries through its abilities to support scene dynamism, reconstruct subpaths generically and operate without being restricted by the necessity of predefined animation paths. The tests that emulate Optis' projects evinced the capabilities of the path manipulation algorithm for the automotive and interior redesign sectors. These results led to discussions with Optis regarding the integration of the path manipulation algorithm with the in-house rendering software.

References

Áfra, A.T., 2012. *Incoherent ray tracing without acceleration structures*. Proceedings of Eurographics 2012 short papers, 97–100.

Aila, T. and Karras, T., 2010. *Architecture considerations for tracing incoherent rays*. HPG '10 proceedings of the conference on high performance graphics, 113–122.

Arvo, J. and Kirk, D., 1990. *Particle transport and image synthesis.* ACM SIGGRAPH Computer graphics, 24 (4), 63–66.

Bagher, M. M., Soler, C., Subr, K., Belcour, L. and Holzschuch, N., 2013. *Interactive rendering of acquired materials on dynamic geometry using frequency analysis*. IEEE transactions on visualization and computer graphics, 19 (5), 749–761.

Bauszat, P., Petitjean, V. and Eisemann, E., 2017. *Gradient-domain path reusing*. ACM transactions on graphics, 36 (6), 229:1–229:9.

Bekaert, P., Sbert, M. and Halton, J., 2002. *Accelerating path tracing by reusing paths*. EGRW '02 proceedings of the 13th Eurographics workshop on rendering, 125–134.

Bikker, J. and van Schijndel, J., 2013. *The Brigade renderer: a path tracer for real-time games*. International journal of computer games technology, 2013, 1–14.

Boulos, S., Edwards, D., Lacewell, J. D., Kniss, J., Kautz, J., Shirley, P. and Wald, I., 2007. *Packet-based Whitted and distribution ray tracing*. Proceedings of graphics interface 2007, 177–184.

CGTrader, 2017. *3D Models* [online]. Available from: https://www.cgtrader.com/3d-models [Accessed 7 September 2017].

Chapman, J., Calvert, T. W. and Dill, J., 1991. *Spatio-temporal coherence in ray tracing*. Proceedings of graphics interface '91, 101–108.

Clarberg, P., Jarosz, W., Akenine-Möller, T. and Jensen, H. W., 2005. *Wavelet importance sampling: efficiently evaluating products of complex functions*. ACM transactions on graphics, 24 (3), 1166–1175.

Cook, R. L., Porter, T. and Carpenter, L., 1984. *Distributed ray tracing*. ACM SIGGRAPH Computer graphics, 18 (3), 137–145.

Cornell University, 2017. *Cornell box data* [online]. Ithaca: Cornell University. Available from: https://www.graphics.cornell.edu/online/box/data.html [Accessed 30 August 2017].

Dachsbacher, C. and Stamminger, M., 2005. *Reflective shadow maps*. Proceedings of the 2005 symposium on interactive 3D graphics and games, 203–231.

Dachsbacher, C., Stamminger, M., Drettakis, G. and Durand, F., 2007. *Implicit visibility and antiradiance for interactive global illumination*. ACM transactions on graphics, 26 (3), 61:1–61:10.

Damez, C., Dimitriev, K. and Myszkowski, K., 2003. *State of the art in global illumination for interactive applications and high-quality animations*. Computer graphics forum, 22 (1), 55–77.

Davidovič, T., Křivánek, J., Hašan, M. and Slusallek, P., 2014. *Progressive light transport simulation on the GPU: survey and improvements*. ACM transactions on graphics, 33 (3), 29:1–29:19.

Dutré, P. and Willems, Y. D., 1995. *Importance-driven Monte Carlo light tracing*. EGRW '95 proceedings of the 5th Eurographics workshop on rendering. 188–197.

Eisenacher, C., Nichols, G., Selle, A. and Burley, B., 2013. *Sorted deferred shading for production path tracing*. EGSR '13 proceedings of the 24th Eurographics symposium on rendering, 32 (4), 125–132.

Georgiev, I., Křivánek, J., Davidovič, T. and Slusallek, P., 2012. *Light transport simulation with vertex connection and merging*. ACM transactions on graphics, 31 (6), 192:1–192:10.

Goral, C. M., Torrance, K. E., Greenberg, D. P. and Battaile, B., 1984. *Modelling the interaction of light between diffuse surfaces*. ACM SIGGRAPH Computer graphics, 18 (3), 213–222.

Günther, J., 2014. *Ray tracing of dynamic scenes*. Dissertation (PhD). Saarland University.

Guntury, S. and Narayanan, P. J., 2010. *Ray tracing dynamic scenes with shadows on the GPU*. Proceedings of the 10th Eurographics conference on parallel graphics and visualization, 27–34.

Guntury, S. and Narayanan, P. J., 2012. *Raytracing dynamic scenes on the GPU using grids*. IEEE transactions on visualization and computer graphics, 18 (1), 5–16.

Hachisuka, T., Ogaki, S. and Jensen, H. W., 2008. *Progressive photon mapping*. ACM transactions on graphics, 27 (5), 130:1–130:8.

Hachisuka, T. and Jensen, H. W., 2009. *Stochastic progressive photon mapping*. ACM transactions on graphics, 28 (5), 141:1–141:8.

Hachisuka, T. and Jensen, H. W., 2010. *Parallel progressive photon mapping on GPUs*. Proceedings of ACM SIGGRAPH Asia '10 Technical Sketches, article no. 54.

Hachisuka, T., Pantaleoni, J. and Jensen, H. W., 2012. *A path space extension for robust light transport simulation*. ACM transactions on graphics, 31 (6), 191:1–191:10.

Hall, R., 1989. *Illumination and color in computer generated imagery*. New York; London: Springer-Verlag.

Hanrahan, P., Salzman, D. and Aupperle, L., 1991. *A rapid hierarchical radiosity algorithm*. ACM SIGGRAPH Computer graphics, 25 (4), 197–206.

Hašan, M., Pellacini, F. and Bala, K., 2007. *Matrix row-column sampling for the many-light problem*. ACM transactions on graphics, 26 (3), 26:1–26:10.

Hašan, M., Křivánek, J., Walter, B. and Bala, K., 2009. *Virtual spherical lights for many-light rendering of glossy scenes*. ACM transactions on graphics, 28 (5), 143:1–143:6.

Havran, V., Damez, C., Myszkowski, K. and Seidel, H.-P., 2003. *An efficient spatio-temporal architecture for animation rendering*. Rendering techniques 2003: 14th Eurographics workshop on rendering, 106–117.

Hecht, E., 2002. *Optics*. 4th international ed. San Francisco; London: Addison-Wesley.

Heckbert, P.S., 1990. *Adaptive radiosity textures for bidirectional path tracing*. ACM SIGGRAPH Computer graphics, 24 (4), 145–154.

Hedman, P., Karras, T. and Lehtinen, J., 2016. *Sequential Monte Carlo instant radiosity*. Proceedings of the 20th ACM SIGGRAPH symposium on interactive 3D graphics and games, 121–128.

Herzog, R., Havran, V., Kinuwaki, S., Myszkowski, K. and Seidel, H.-P., 2007. *Global illumination using photon ray splatting*. Computer graphics forum, 26 (3), 503–513.

Immel, D. S., Cohen, M. F. and Greenberg, D. P., 1986. *A radiosity method for non-diffuse environments*. ACM SIGGRAPH Computer graphics, 20 (4), 133–142.

Iones, A., Krupkin, A., Sbert, M. and Zhukov, S., 2003. *Fast, realistic lighting for video games*. IEEE computer graphics and applications, 23 (3), 54–64.

Jakob, W. and Marschner, S., 2012. *Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport*. ACM transactions on graphics, 31 (4), 58:1–58:13.

Jensen, H. W., 2001. *Realistic image synthesis using photon mapping*. Canada: A. K. Peters Ltd.

Jurohi, 2006. *BSDF05_800.png* [image]. San Francisco: Wikimedia Foundation. Available from https://en.wikipedia.org/wiki/File:BSDF05_800.png [Accessed 19 April 2017].

Kajiya, J. T., 1986. *The rendering equation*. ACM SIGGRAPH Computer graphics, 20 (4), 143–150.

Kautz, J. and McCool, M. D., 1999. *Interactive rendering with arbitrary BRDFs using separable approximations*. EGWR '99 proceedings of the 10th Eurographics workshop on rendering, 247–260.

Kautz, J., Sloan, P.-P. and Snyder, J., 2002. *Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics*. EGRW '02 proceedings of the 13th Eurographics workshop on rendering, 291–296.

Keller, A., 1997. *Instant radiosity*. SIGGRAPH '97 proceedings of the 24th annual conference on computer graphics and interactive techniques, 49–56.

Kettunen, M., Manzi, M., Aittala, M., Lehtinen, J., Durand, F. and Zwicker, M., 2015. *Gradient-domain path tracing*. ACM transactions on graphics, 34 (4), 123:1–123:13.

Knecht, M., 2009. *Real-time global illumination using temporal coherence*. Thesis (MSc). Vienna University of technology.

Lafortune, E. P. and Willems, Y. D., 1993. *Bi-directional path tracing*. Proceedings of 3rd international conference on computational graphics and visualization techniques, 145–153.

Lai, Y. C., 2010. *Photorealistic animation rendering with population Monte Carlo energy redistribution*. Dissertation (PhD). University of Wisconsin at Maddison.

Lauterbach, C., Yoon, S.-E., Tuft, D. and Manocha, D., 2006. *RT-DEFORM: interactive ray tracing of dynamic scenes using BVHs*. Proceedings of the 2006 IEEE symposium on interactive ray tracing, 39–46.

Manzi, M., Kettunen, M., Aittala, M., Lehtinen, J., Durand, F. and Zwicker, M., 2015. *Gradient-domain bidirectional path tracing*. Proceedings of Eurographics symposium on rendering – experimental ideas & implementations, 1–10.

Manzi, M., Kettunen, M., Durand, F., Zwicker, M. and Lehtinen, J., 2016. *Temporal gradient-domain path tracing*. ACM transactions on graphics, 35 (6), 246:1–246:9.

Martin, W., Shirley, P., Parker, S., Thompson, W. and Reinhard, E., 2002. *Temporally coherent interactive ray tracing*. Journal of graphics tools, 7 (2), 41–48.

Matusik, W., Pfister, H., Brand, M. and McMillan, L., 2003. *A data-driven reflectance model*. ACM transactions on graphics, 22 (3), 759–769.

McGuire, M. and Luebke, D., 2009. *Hardware-accelerated global illumination by image space photon mapping*. HPG '09 proceedings of the conference on high performance graphics 2009, 77–89.

Méndez-Feliu, A., Sbert, M. and Szirmay-Kalos, L., 2006. *Reusing frames in camera animation*. Journal of WSCG, 14 (1–3), 97–104.

Méndez-Feliu, A. and Sbert, M., 2006. *Efficient rendering of light and camera animation for navigating a frame array*. Computer animation and social agents, 1–8.

Mobley, C., Boss, E. and Roesler, C., 2017. *Ocean optics web book* [online].

Mora, B., 2011. *Naive ray-tracing: a divide-and-conquer approach*. ACM transactions on graphics, 30 (5), 117:1–117:12.

Nabata, K., Iwasaki, K., Dobashi, Y. and Nishita, T., 2013. *Efficient divide-and-conquer ray tracing using ray sampling*. HPG '13 proceedings of the 5th high-performance graphics conference, 129–135.

Nah, J.-H., Kim, J.-W., Park, J., Lee, W.-J., Park, J.-S., Jung, S.-Y., Park, W.-C., Manocha, D. and Han, T.-D., 2015. *HART: a hybrid architecture for ray tracing animated scenes*. IEEE transactions on visualization and computer graphics, 21(3), 389–401.

Ng, R., Ramamoorthi, R. and Hanrahan, P., 2004. *Triple product wavelet integrals for all-frequency relighting*. ACM transactions on graphics, 23 (3), 477–487.

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W. and Limperis, T., 1977. *Geometrical considerations and nomenclature for reflectance*. United States of America: National Bureau of Standards. Monograph 160.

Nishita, T., Sirai, T., Tadamura, K. and Nakamae, E., 1993. *Display of the earth taking into account atmospheric scattering*. Proceedings of the 20th annual conference on computer graphics and interactive techniques, 175–182.

Novák, J., Havran, V. and Dachsbacher, C., 2010. *Path regeneration for interactive path tracing*. Proceedings of Eurographics 2010 short papers, 61–64.

NVIDIA, 2017. *NIVIDIA OptiX ray tracing engine* [online]. Available from: https://developer.nvidia.com/optix [Accessed 2 March 2017].

Pajot, A., Barthe, L., Paulin, M. and Poulin, P., 2011. *Combinatorial bidirectional path-tracing for efficient hybrid CPU/GPU rendering*. Computer graphics forum, 30 (2), 315–324.

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A. and Stich, M., 2010. *OptiX: a general purpose ray tracing engine*. ACM transactions on graphics, 29 (4), 66:1–66:13.

Pérard-Gayot, A., Kalojanov, J. and Slusallek, P., 2017. *GPU ray tracing using irregular grids*. Computer graphics forum, 36 (2), 477–486.

Pharr, M. and Humphreys, G., 2004. *Physically based rendering: from theory to implementation*. San Francisco, California, USA: Morgan Kaufmann Publishers Inc.

Popov, S., Ramamoorthi, R., Durand, F. and Drettakis G., 2015. *Probabilistic connections for bidirectional path tracing*. EGSR '15 proceedings of the 26th Eurographics symposium on rendering, 34 (4), 75–86.

Ravichandran, S. and Narayanan, P. J., 2013. *Parallel divide and conquer ray tracing*. Proceedings of SIGGRAPH Asia 2013 technical briefs, 30:1–30:4.

Reenskaug, T., 1979. *Models-views-controllers* [online]. Palo Alto: Xerox Palo Alto Research Laboratory.

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C. and Kautz, J., 2008. *Imperfect shadow maps for efficient computation of indirect illumination*. ACM transactions on graphics, 27 (5), 129:1–129:8.

Ritschel, T., Dachsbacher, C., Grosch, T. and Kautz, J., 2012. *The state of the art in interactive global illumination*. Computer graphics forum, 31 (1), 160–188.

Rousselle, F., Jarosz, W. and Novák, J., 2016. *Image-space control variates for rendering*. ACM transactions on graphics, 35 (6), 169:1–169:12.

Rusinkiewicz, S. M., 1998*. A new change of variables for efficient BRDF representation*. Rendering techniques 1998: proceedings of the Eurographics workshop, 11–22.

Sanders, J. and Kandrot, E., 2012. *CUDA by example: An introduction to general-purpose GPU programming*. 4th edition. United States of America: Addison-Wesley.

Sbert, M. and Castro, F., 2004. *Reuse of paths in final gathering step with moving light sources*. The 4th international conference on computational science, 189–196.

Sbert, M., Castro, F. and Halton, J., 2004a. *Reuse of paths in light source animation*. Proceedings of computer graphics international, 532–535.

Sbert, M., Szécsi, L. and Szirmay-Kalos, L., 2004b. *Real-time light animation*. Computer graphics forum, 23 (3), 291–299.

Segovia, B., Iehl, J. C., Mitanchey, R. and Péroche, B., 2006. *Bidirectional instant radiosity*. EGSR '06 proceedings of the 17th Eurographics conference on rendering techniques, 389–397.

Segovia, B., Iehl, J. C. and Péroche, B., 2007. *Metropolis instant radiosity*. Computer graphics forum, 26 (3), 425–434.

Šik, M., Otsu, H., Hachisuka, T. and Křivánek, J., 2016. *Robust light transport simulation via metropolised bidirectional estimators*. ACM transactions on graphics, 35 (6), 245:1–245:12.

Sloan, P.-P., Kautz, J. and Snyder, J., 2002. *Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments*. ACM transactions on graphics, 21 (3), 527–536.

Smits, B., Arvo, J. and Greenberg, D., 1994. *A clustering algorithm for radiosity in complex environments*. SIGGRAPH '94 proceedings of the 21st annual conference on computer graphics and interactive techniques, 435–442.

Spencer, B. and Jones, M. W., 2013. *Progressive photon relaxation*. ACM transactions on graphics, 32 (1), 7:1–7:11.

Tawara, T., Myszkowski, K., Dmitriev, K., Havran, V., Damez, C. and Seidel, H.-P., 2004. *Exploiting temporal coherence in global illumination*. SCCG '04 proceedings of the 20th spring conference on computer graphics, 23–33.

Tokuyoshi, Y. and Ogaki, S., 2012. *Real-time bidirectional path tracing via rasterization*. Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games, 183–190.

Van Antwerpen, D., 2011a. *Improving SIMD efficiency for parallel Monte Carlo light transport on the GPU*. HPG '11 proceedings of the ACM SIGGRAPH symposium on high performance graphics, 41–50.

Van Antwerpen, D., 2011b. *Unbiased physically based rendering on the GPU*. Thesis (MSc). Delft University of Technology.

Veach, E. and Guibas, L., 1994. *Bidirectional estimators for light transport*. Proceedings of Eurographics rendering workshop, 147–162.

Veach, E. and Guibas, L. J., 1995. *Optimally combining sampling techniques for Monte Carlo rendering.* SIGGRAPH '95 proceedings of the 22nd annual conference on computer graphics and interactive techniques, 419–428.

Veach, E. and Guibas, L. J., 1997. *Metropolis light transport*. SIGGRAPH '97 proceedings of the 24th annual conference on computer graphics and interactive techniques, 65–76.

Veach, E., 1998. *Robust Monte Carlo methods for light transport simulation*. Dissertation (PhD). Stanford University.

Vorba, J., Karlík, O., Šik, M., Ritschel, T. and Křivánek, J., 2014. *On-line learning of parametric mixture models for light transport simulation*. ACM transactions on graphics, 33 (4), 101:1–101:11.

Wald, I., Kollig, T., Benthin, C., Keller, A. and Slusallek, P., 2002. *Interactive global illumination using fast ray tracing*. EGRW '02 proceedings of the 13th workshop on rendering, 15–24.

Wald, I., Benthin, C. and Slusallek, P., 2003. *Distributed interactive ray tracing of dynamic scenes*. Proceedings of the 2003 IEEE symposium on parallel and large-data visualization and graphics, 77–85.

Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S. G. and Shirley, P., 2007. *State of the art in ray tracing animated scenes*. Eurographics 2007 state of the art reports, 89–116.

Wald, I., Woop, S., Benthin, C., Johnson, G. S. and Ernst, M., 2014. *Embree: a kernel framework for efficient CPU ray tracing*. ACM transactions on graphics, 33 (4), 143:1–143:8.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M. and Greenberg, D. P., 2005. *Lightcuts: a scalable approach to illumination*. ACM transactions on graphics, 24 (3), 1098–1107.

Wang, R., Tran, J. and Luebke, D., 2005. *All-frequency interactive relighting of translucent objects with single and multiple scattering*. ACM transactions on graphics, 24 (3), 1202–1207.

Whitted, T., 1980. *An improved illumination model for shaded display*. Communications of the ACM, 23 (6), 343–349.

Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C. and Yoon, S.-E., 2015. *Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering.* Computer graphics forum, 34 (2), 667–681.