2022

Correlations and Reuse for Fast and Accurate Physically Based Light Transport

Benedikt Bitterli
Benedikt.M.Bitterli.GR@Dartmouth.edu

Follow this and additional works at: https://digitalcommons.dartmouth.edu/dissertations

Part of the Graphics and Human Computer Interfaces Commons

Recommended Citation
Bitterli, Benedikt, "Correlations and Reuse for Fast and Accurate Physically Based Light Transport" (2022). Dartmouth College Ph.D Dissertations. 77.
https://digitalcommons.dartmouth.edu/dissertations/77

This Thesis (Ph.D.) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Ph.D Dissertations by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.
CORRELATIONS AND REUSE FOR FAST AND ACCURATE PHYSICALLY BASED LIGHT TRANSPORT

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in
Computer Science
by BENEDIKT BITTERLI
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire
August 2021

______________________________
(chair) Wojciech Jarosz (Dartmouth College)

______________________________
Bo Zhu (Dartmouth College)

______________________________
Peter Winkler (Dartmouth College)

______________________________
Steve Marschner (Cornell University)

______________________________
Jaakko Lehtinen (Aalto University)

F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies
To Martin and Christiane
ABSTRACT

Light transport is the study of the transfer of light between emitters, surfaces, media and sensors. Fast simulations of light transport play a pivotal role in photo-realistic image synthesis, and find many applications today, including predictive manufacturing, machine learning, scientific visualization and the movie industry. In order to accurately reproduce the appearance of real scenes, light transport must closely approximate the physical laws governing the flow of light. Physically based rendering is a set of principles for codifying these laws into a mathematical model, and is the predominant rendering methodology today.

The representational power of this model is limited to the effects it chooses to capture. Simultaneously, simulating the model is an additional source of approximation error: The predominant solution framework in use today—Monte Carlo integration—produces the exact image predicted by the model typically only in the limit of infinite computation; at any finite time, an image contaminated with noise is obtained.

In this dissertation, we are concerned with improving the accuracy of physically based light transport. We achieve this both by improving the representational power of the model, and by making the rendering algorithms more efficient, leading to lower error at any given computational budget. In particular, we will investigate correlations and reuse: On the one hand, prevalent models in rendering assume natural processes to arise from random, independent events, and simulate them as such. We will show that for participating media—such as clouds, fog, or smoke—this assumption does not hold, and we introduce an augmented model that can faithfully represent such correlations. On the other hand, the types of solutions that satisfy the
rendering problem show a great deal of correlation. Because all pixels in an image view the same scene, the mathematical problems to be solved are greatly interrelated. Where naive rendering algorithms treat each pixel in isolation, we will focus on reusing the same computation over many pixels, exploiting the natural correlations present and thus amortizing computational effort. We improve over prior work by leveraging additional insights about the structure of the rendering problem to allow a greater amount of reuse, and thus efficiency.
No book, however good, can survive a hostile reading.

— Orson Scott Card

ACKNOWLEDGMENTS

A PhD is a long and sometimes arduous journey, and I would go remiss without thanking the many people that have accompanied me on this adventure.

I owe an immense debt of gratitude to my family and friends, whose ceaseless outpouring of support and encouragement have allowed me to accomplish things that sometimes felt impossible. I would like to thank Andrew Chin, Aaron Griffith, Alex Headley and Lucas Hereld for their friendship, the many nights of entertainment and camaraderie over the past five years; they have been a source of joy on days that otherwise seemed bleak, and it makes a world of difference to hear “Congratulations” after completing a major life chapter. I would like to extend my sincerest thanks to Dario Seyb for being a good friend and always being there in times of need, and Zack Misso for being an outstanding human being and labmate during my time at Dartmouth. I would like to thank Maria for always fearlessly reaching out, and I greatly appreciated her many letters over the years. A deep thanks also goes to Tabitha for always being a great source of encouragement and levity.

I would like to express my sincerest gratitude to my advisor, Wojciech Jarosz, for his many years of mentorship, guidance and inspiration. Starting even years before my PhD, his deep insight, knowledge and talent have been both an incredible resource and inspiration, and I owe to him much of what makes me a researcher today. His door was always open, and he never hesitated to make time for me. His unrelenting pursuit of good research and writing are ideals to strive for, and I am grateful to have had the opportunity to study with him.
I am grateful to my many mentors over the years. I have deep appreciation for Brent Burley for his encouragement and mentorship; his bottomless knowledge, foresight and vision have touched the field in more ways than one, and I fondly look back on our conversations. Deep thanks also go to Jan Novák and Wenzel Jakob, who are outstanding researchers and have been a great source of inspiration over the years. I count myself fortunate to have been able to learn from them, and I have greatly enjoyed their collaboration. I am especially grateful to Aaron Lefohn for his boundless enthusiasm that has given new meaning to doing graphics research, and for his support during the last two years. I have greatly enjoyed working with him, and am excited to continue to do so.

I would like to thank Matt Pharr, Peter Shirley, Chris Wyman, Thomas Müller and Alex Evans for our many inspiring and interesting talks. They continue to be a source of excitement and motivation to push forward in new directions. A special thanks goes to Steve Marschner, whose thoughtful comments have been of immeasurable help, and whose deep insight is an ideal to strive toward; I greatly appreciate his willingness to take time for me.

I am deeply obliged to my parents Martin and Christiane, who have unwaveringly supported me in my passions from day one, even through the many twists and turns and over great distances. They could always see potential in me, even when I could not.

Finally, I thank my wonderful partner Rachel for her neverending love, patience and support she afforded me during my many years at Dartmouth—through the warm Hanover summers and dark New England winters, both literal and metaphorical, she has always been there for me.

My research has been supported by generous grants from Activision, Microsoft Azure, the National Science Foundation (CNS-1205521, IIS-1812796) and NVIDIA (both directly, and via the NVIDIA graduate fellowship).
Parts of this dissertation are based on published works that I have co-authored with others. I reproduce parts of these works, including text and figures, in this dissertation with explicit permission of the co-authors. My contribution to each paper is listed below:

- **Chapter 4** is based on the work that appears in the paper:
  

  I was the primary investigator and author of this paper.

- **Chapter 5** is based on work that appears in the following papers:
  

I was the primary investigator and author of the first paper, and co-author of the second paper. Chapter 5 is predominantly based on the first paper. In Section 5.3, we briefly summarize the mathematical framework of the second paper. Of the second paper, I only claim the reproduced section as my contribution.

- **Chapter 6** is based on work that appears in the paper:


  I was the primary investigator and author of this paper.

- **Chapter 7** is based on work that appears in the paper:


  I was the primary investigator and author of this paper.

- **Chapter 8** is based on work that appears in the paper:


  I was the primary investigator and author of this paper.

  *Appendix A.1* shows an additional derivation of non-exponential transport on discrete particles. This derivation was contributed by Stever Marschner and is re-
produced here for completeness, with permission. I do not claim authorship of this section.
# CONTENTS

1 **INTRODUCTION**  
1.1 Organization of the Dissertation ................................................. 4

2 **BACKGROUND**  

2.1 **FUNDAMENTALS OF LIGHT TRANSPORT**  
2.1 Notation .......................................................................................... 9  
2.2 Scene Representation ................................................................. 10  

2.1.1 Surfaces .................................................................................... 11  
2.1.2 Participating Media .................................................................... 12  
2.3 Radiometry .................................................................................... 14  
2.4 Scattering Models .......................................................................... 16  
2.5 Balance on Surfaces ...................................................................... 18  
2.6 Balance in Participating Media ..................................................... 19  
2.7 Combined Balance ........................................................................ 21  
2.8 Cartesian formulation .................................................................... 22  
2.9 The Path Integral and the Measurement Equation  
2.9.1 Measurement Equation ............................................................. 25  
2.9.2 A Note on Notation ................................................................. 28

3 **INTEGRATION METHODS**  
3.1 Basics of Monte Carlo Integration .................................................. 30  

3.1.1 Variance ................................................................................... 30  
3.1.2 Primary Estimators .................................................................... 31  
3.1.3 Bias and Consistency ............................................................... 32  
3.1.4 Importance Sampling ............................................................... 34  

3.1.4.1 The Inversion Method ......................................................... 34  
3.1.4.2 Resampled Importance Sampling ....................................... 35  
3.1.4.3 Other Importance Sampling Techniques ............................ 35  
3.1.5 Multiple Importance Sampling ................................................. 36  
3.2 Monte Carlo Rendering Algorithms .............................................. 38  

3.2.1 Light-, Path- and Bidirectional Path Tracing ............................. 38  
3.2.2 Photon Mapping ....................................................................... 39  
3.2.3 Markov Chain Monte Carlo .................................................... 42
## II METHODS

4 A radiative transfer framework for non-exponential media

4.1 The case for non-exponential transport ............................................. 53
4.2 Non-exponentiality is Non-trivial ....................................................... 56
4.3 A Non-Exponential Path Integral ....................................................... 57
  4.3.1 Reciprocity .................................................................................. 66
  4.3.2 Heterogeneity ............................................................................. 67
  4.3.3 Related Non-Exponential Models ............................................... 70
4.4 Modeling non-exponential attenuation ............................................... 73
  4.4.1 Transmittance via directly designed free-flight PDFs .................... 74
  4.4.2 Data-driven transmittance via ensemble averaging ....................... 77
  4.4.3 Probabilistic ensemble-averaging ............................................... 80
  4.4.4 Closed-form average transmittance in fractal media ..................... 81
  4.4.5 Combining macro- and micro- properties ..................................... 86
4.5 Implementation .................................................................................. 88
  4.5.1 Unbiased Delta Tracking ............................................................ 90
4.6 Results .............................................................................................. 91
4.7 Discussion ......................................................................................... 97
  4.7.1 Contributions .............................................................................. 97
  4.7.2 Limitations ................................................................................ 98
  4.7.3 Follow-up Work ......................................................................... 101

5 Higher-dimensional photon samples for volumetric light transport

5.1 Beams as a Limit Process ................................................................. 103
  5.1.1 Prior Density Estimators .......................................................... 105
5.2 nD photons as a limit process ........................................................ 109
5.3 nD Photons as Solutions in an Extended Path Space ....................... 114
5.4 Transmittance Estimators and “Long” vs. “Short” Beams ................ 117
5.5 Theoretical Error Analysis .............................................................. 119
  5.5.1 Analysis Setup .......................................................................... 119
  5.5.2 Relation to Volumetric Density Estimators .................................. 121
  5.5.3 Error Derivation ....................................................................... 122
  5.5.4 Error Comparison and Discussion ............................................. 123
8.1 Problem Setting ........................................... 185
8.2 Resampled Importance Sampling .......................... 186
8.3 Reservoir Resampling .................................... 188
8.4 Reservoir Reuse ........................................... 191
8.5 Eliminating Bias .......................................... 196
  8.5.1 The RIS PDF ........................................ 197
  8.5.2 Analyzing the RIS Weight .......................... 199
  8.5.3 The Expected RIS Weight .......................... 200
  8.5.4 Unbiased RIS ....................................... 203
  8.5.5 A Practical Algorithm for Unbiased Reuse ......... 205
8.6 Design and Implementation Choices ....................... 206
8.7 Results ................................................... 210
8.8 Discussion ............................................... 213
  8.8.1 Contributions ....................................... 214
  8.8.2 Limitations ......................................... 215
  8.8.3 Follow-up Work .................................... 216
8.9 Improved Reservoir Resampling .......................... 218
  9.1 Pairwise MIS for Robust Reservoir Reuse ............... 219
  9.1.1 RIS as a Secondary MC Estimator .................. 220
  9.1.2 A Multiple Importance Sampled RIS Estimator .... 221
  9.1.3 An efficient MIS heuristic ........................ 222
  9.1.4 Results ............................................ 225
  9.2 Scene-Space Reservoir Reuse ........................... 227
    9.2.1 Algorithm Overview ................................ 228
    9.2.2 Aggregate Target Distributions .................. 232
      9.2.2.1 Geometry Term ................................ 233
      9.2.2.2 Scattering Function .......................... 235
      9.2.2.3 Combined Aggregate Target Distribution .... 236
    9.2.3 Reservoir Reuse with Aggregate Reservoirs ....... 236
    9.2.4 Choice of Data Structure ......................... 238
      9.2.4.1 Tree Structure Overview ........................ 239
      9.2.4.2 Tree Maintenance ................................ 240
    9.2.5 Implementation Details and Results ................ 241
      9.2.5.1 Implementation Details ........................ 242
      9.2.5.2 Results ....................................... 242
  9.3 Discussion, Analysis and Future Work ................... 245
    9.3.1 Contributions .................................... 246
CONTENTS

III CONCLUSION

10 CONCLUSION AND OUTLOOK

IV APPENDIX

A ADDITIONAL DERIVATIONS

BIBLIOGRAPHY
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Illustration of surfaces and media in a scene</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Radiance measured on a hypothetical surface patch and direction cone</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Demonstration of different types of scattering in media and surfaces</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Illustration of the direction/distance and Cartesian form</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>An example path and corresponding throughput terms</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison between concatenation of paths, and the extended path in photon mapping</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of discrete and continuous models of media</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustration of the segment decomposition approximation</td>
<td>61</td>
</tr>
<tr>
<td>4.3</td>
<td>The four transport functions used in our transmittance model</td>
<td>63</td>
</tr>
<tr>
<td>4.4</td>
<td>Intuitive interpretation of our heterogeneity model</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>Phenomenological transmittances rendered in our transport framework</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>1D transects, line-averaged density and histograms of perlin noise</td>
<td>79</td>
</tr>
<tr>
<td>4.7</td>
<td>Relationships between different colors of noise</td>
<td>82</td>
</tr>
<tr>
<td>4.8</td>
<td>Ensemble-averaged transmittances/PDFs arising from Gamma-distributed fractal</td>
<td>85</td>
</tr>
<tr>
<td>4.9</td>
<td>Interpretation of fractal noise in our heterogeneity model</td>
<td>87</td>
</tr>
<tr>
<td>4.10</td>
<td>Comparing regular tracking and unbiased delta tracking in our method</td>
<td>90</td>
</tr>
<tr>
<td>4.11</td>
<td>Comparison of bidirectional methods rendering our light transport model</td>
<td>91</td>
</tr>
<tr>
<td>4.12</td>
<td>Complex cloud rendered with a parametric transmittance</td>
<td>92</td>
</tr>
<tr>
<td>4.13</td>
<td>Homogeneous medium rendered with different transmittances</td>
<td>93</td>
</tr>
<tr>
<td>4.14</td>
<td>Heterogeneous cloud rendered with transmittance derived from fractal noise</td>
<td>95</td>
</tr>
<tr>
<td>4.15</td>
<td>Plots of a Monte Carlo simulation comparing our model to ground truth</td>
<td>96</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.16</td>
<td>Non-physical Tr resulting from Davis et al.’s model</td>
<td>100</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of generalizing prior work to nD photons</td>
<td>103</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of transmittance estimators</td>
<td>118</td>
</tr>
<tr>
<td>5.3</td>
<td>Our canonical error analysis setup</td>
<td>120</td>
</tr>
<tr>
<td>5.4</td>
<td>Plot of rRMSEs of different estimator types</td>
<td>123</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of renderings of our method and prior work at equal render time</td>
<td>133</td>
</tr>
<tr>
<td>5.6</td>
<td>Continuation of Fig. 5.5</td>
<td>134</td>
</tr>
<tr>
<td>6.1</td>
<td>Illustration of multiplexed primary sample space</td>
<td>141</td>
</tr>
<tr>
<td>6.2</td>
<td>Non-ambiguities arising from non-injective mappings</td>
<td>147</td>
</tr>
<tr>
<td>6.3</td>
<td>1D example showing the correctness of RJMLT</td>
<td>155</td>
</tr>
<tr>
<td>6.4</td>
<td>Plots of MSE and standard deviations of RJMLT and MMLT</td>
<td>158</td>
</tr>
<tr>
<td>6.5</td>
<td>Average perturbation acceptance rate in RJMLT and MMLT</td>
<td>159</td>
</tr>
<tr>
<td>6.6</td>
<td>Renderings comparing results of RJMLT and MMLT at equal render time</td>
<td>160</td>
</tr>
<tr>
<td>6.7</td>
<td>Continuation of Fig. 6.6</td>
<td>161</td>
</tr>
<tr>
<td>7.1</td>
<td>Behavior of MC and MCMC methods in a toy example</td>
<td>167</td>
</tr>
<tr>
<td>7.2</td>
<td>Illustration of our outlier detector</td>
<td>174</td>
</tr>
<tr>
<td>7.3</td>
<td>Comparison of renderings obtained with path tracing, SMLT, ERPT and RJMLT at equal render time</td>
<td>178</td>
</tr>
<tr>
<td>7.4</td>
<td>Plots showing MSE and its standard deviation of SMLT and prior work</td>
<td>179</td>
</tr>
<tr>
<td>8.1</td>
<td>Evolution of reservoir resampling with increasing M</td>
<td>191</td>
</tr>
<tr>
<td>8.2</td>
<td>High-level overview of ReSTIR</td>
<td>192</td>
</tr>
<tr>
<td>8.3</td>
<td>Comparing reservoir resampling to repeated spatial reuse</td>
<td>193</td>
</tr>
<tr>
<td>8.4</td>
<td>Comparing spatial reuse alone to spatiotemporal reuse</td>
<td>194</td>
</tr>
<tr>
<td>8.5</td>
<td>Comparing spatiotemporal reuse alone to spatiotemporal with visibility reuse</td>
<td>195</td>
</tr>
<tr>
<td>8.6</td>
<td>1D example of biased RIS, naive debiasing and MIS in resampling</td>
<td>202</td>
</tr>
<tr>
<td>8.7</td>
<td>Comparison of equal-time renderings of ReSTIR and reservoir resampling</td>
<td>209</td>
</tr>
<tr>
<td>8.8</td>
<td>Comparison of equal-time renderings of ReSTIR and the algorithm of Moreau et al.</td>
<td>211</td>
</tr>
<tr>
<td>8.9</td>
<td>Evolution of the RMAE of ReSTIR and prior work</td>
<td>212</td>
</tr>
<tr>
<td>8.10</td>
<td>Equal-time comparison of ReSTIR and reservoir resampling at 1s of render time</td>
<td>213</td>
</tr>
</tbody>
</table>
Figure 9.1  Comparison of the ReSTIR to pairwise ReSTIR ................................. 226
Figure 9.2  Continuation of Fig. 9.1 ................................................................. 227
Figure 9.3  Illustration of the scene-space extension of ReSTIR ....................... 230
Figure 9.4  Illustration of the aggregate target function ................................. 232
Figure 9.5  Visualization of the octree data structure ...................................... 243
Figure 9.6  Comparison of the octree-based ReSTIR to pairwise ReSTIR .......... 244
Figure 9.7  Visualization of various spatial reuse patterns ............................... 248
Figure 9.8  Demonstration of visual artifacts appearing in hierarchical reuse 249
Figure 9.9  Using Sudoku tiles to obtain tree interleaving .............................. 251
Figure 9.10 RMAE plots and equal-time renderings of randomized and
              hierarchical reuse ................................................................. 252
Figure 9.11 Histogram visualization of the sample choice in RIS ..................... 254
Figure 9.12 Sample clumping due to reservoir overuse ................................... 256
Figure A.1  Variance of control variates and naive MC sampling .................... 271

LIST OF TABLES

Table 2.1  List of quantities and their units .................................................... 15
Table 4.1  Table of PDFs, CDFs and CFs for various distributions ..................... 76
Table 4.2  Sampling routines of statistical distributions .................................... 89
Table 5.1  Ablation study of various optimizations in our method ...................... 127
Table 5.2  Table of effective speedups of our new estimators .......................... 131
INTRODUCTION

Computer graphics is the study of generating and manipulating images with the aid of computers. It is driven by the fundamental human desire to visually represent and approximate experiences, both those rooted in reality and those imagined entirely. The act of depicting scenes in images goes back tens of thousands of years in human history beginning with cave paintings made in charcoal, and has since evolved to make use of a stunning array of media and techniques. Computers are merely the latest medium used to make images.

Of the many movements in the arts, realism concerns itself with the accurate representation of the visual appearance of scenes and objects. Depicting realistic images requires a basic understanding of the nature of light, color and geometry, which has been distilled into many heuristics of varying sophistication, such as distant objects appearing smaller than closer objects [41], or landscapes appearing blue with increasing distance from the observer [30].

Physically based rendering formalizes these heuristics into a mathematical model that expresses image formation in terms of the flow of light from emitters to sensors that can be simulated on a computer. The ability to generate (render) realistic images on a computer is of immense interest to many industries today, including movie production, video games and scientific visualization. Of increasing importance are also applications where the end consumer of images is no longer a human, but a computer attempting to solve a task by predicting the appearance of unseen states. This
has enabled entirely new applications in predictive manufacturing and the training of neural networks, for example.

The mathematical model used in physically based rendering is based on the real physical processes underlying imaging, both in the human visual system and artificial sensors. The accuracy of this model is limited both by the physical laws it incorporates—or rather, those it does not—and the algorithms used to simulate the model. While in theory, we could readily obtain images indiscernible from reality if we were to simulate light according to our most complete set of physical laws, doing so would be entirely infeasible in practice due to the computational cost involved.

Because of this, images produced with physically based rendering are only an approximate representation of reality. The approximation error of this methodology comes from two sources: First, the model itself, which only incorporates a simplified subset of our full understanding of light; and second, in the numerical algorithms used to simulate the model. This is because producing solutions predicted by modern rendering models requires solving high-dimensional integration problems. Save for special cases, solutions to these integrals cannot be derived in closed form, and numerical integration methods have become the de facto foundation of modern rendering. Because of the structure and high dimensionality of the integration problem, the most efficient rendering algorithms known today are based on Monte Carlo integration, in which an integral is estimated from random samples of the integrand. An important property of these algorithms is that their accuracy is directly related to the rendering time: The longer the algorithm is allowed to run, the lower its error. Conversely, given a fixed computational budget, any improvement in the efficiency of the algorithm will lead to lower error.

In a sense, all of the components involved in producing an image—the rendering model, the algorithms used to simulate the model, and the hardware used to execute the rendering algorithms—have undergone a form of coevolution, in which the hardware dictates which types of algorithms can be executed efficiently, while
the algorithms available dictate an upper bound to the model complexity that can be supported given a fixed computational budget\(^1\). At the same time, future hardware design is influenced by the needs of existing algorithms and models. A notable example of this is the recent introduction of affordable, consumer-grade \textit{ray-tracing} hardware \cite{ray-tracing-hardware} that provides dedicated circuitry for the ray-tracing operation demanded by modern rendering algorithms.

This interplay of models, algorithms and hardware is what makes computer graphics a continually and rapidly evolving field. Where widespread adoption of physically based rendering was only enabled by progress in computing hardware to begin with, the model and algorithms must now continually adapt to keep up with new hardware capabilities and meet the ever-increasing need for realism and accuracy in renderings.

In this dissertation, we will work towards improving the accuracy of physically based rendering both by modifying the model directly, and by introducing new rendering algorithms that can render existing scenes more efficiently. These faster algorithms allow reaching a desired integration error in less time than existing work, or, in turn, enable more accurate but costlier models to be used given the same computational budget.

In particular, we will investigate \textit{correlations} and \textit{reuse} to derive more efficient algorithms and more accurate models. On the one hand, many models in graphics assume that natural processes are the result of stochastic, but \textit{independent} events, so that they can be simulated easily with algorithms that make only local decisions. For the specific case of volumetric materials—such as clouds, fog, turbid liquids such as milk or even our own skin—we show that this assumption is violated however, and

\footnote{Although computational budgets have increased exponentially over the past few decades, following \textit{Moore’s law} \cite{moore-law}, the computational complexity of rendering algorithms has increased at a commensurate rate, postulated by \textit{Blinn’s law} \cite{blinn-law}, p. 48, which predicts average render times to remain invariant over time.}
they are instead more faithfully represented with an enriched model that can capture long-range correlations between interaction events with the material.

At the same time, the types of integration problems posed by the physically based rendering model themselves show a great deal of correlation. Rendering an image involves solving a great number of interrelated integrals, as all pixels share a view of the same virtual scene. Where naive rendering algorithms solve each integration problem in isolation, we will focus on algorithms that reuse the same computation over many pixels, making use of the natural correlations present between integration problems and thus amortizing computational effort. We will improve over previous work by introducing additional insights about the structure of the rendering problem in question to allow a greater amount of reuse, and thus efficiency.

1.1 Organization of the Dissertation

This dissertation is organized into ten chapters. In Chapter 2, we review the basic rendering model and mathematical laws considered in this dissertation, and introduce the notation used in the following chapters. In Chapter 3, we review existing rendering algorithms used to simulate the model of Chapter 2. We will focus on the methods and algorithms most relevant to our own contributions. The following chapters introduce our novel contributions; we end each chapter with a summary of our contributions and their limitations. In Chapter 4, we introduce our new practical light transport model that can represent correlated interactions with participating media. In subsequent chapters, we introduce new algorithms that could be used to simulate this (and other) models, beginning with a novel extension to photon mapping in Chapter 5 that allows for dramatically increased opportunities for reuse of computation in participating media. We follow in Chapter 6 with a reformulation of a class of Metropolis rendering methods in the framework of reversible jumps, which
will allow us to bridge two thus-far separate rendering domains and introduce a more efficient Metropolis algorithm that exploits the structure of these domains for improved reuse between samples. In Chapter 7, we address the shortcomings of this (and similar) algorithms by only using them selectively where they provide a benefit. In Chapter 8, we consider recent hardware developments and introduce a new rendering algorithm that exploits hardware ray-tracing features while still allowing for significant reuse of computation across pixels. This algorithm is specialized to a particular problem domain and suffers from several deficiencies, and we propose several solutions and extensions to this algorithm in Chapter 9. In Chapter 10, we summarize the contributions of this thesis, its limitations and discuss opportunities for future work.
Part I

BACKGROUND

Madness, and then illumination.
— Orson Scott Card
FUNDAMENTALS OF LIGHT TRANSPORT

The principal goal of rendering algorithms is to generate images of a virtual scene that appear realistic to a human observer. Contemporary rendering algorithms achieve this goal by means of simulating the physical process underlying image formation in the real world, i.e. the transfer of radiative energy between emitters and sensors. This is light transport.

In this chapter, we will review the model of light transport we will use for the rest of this dissertation. The model we use is based on the seminal work of Kajiya [78] and Veach [131]. We will import the most relevant methodology from these works and refer to the original authors for a full discussion.

Like all physical systems, light transport can be described by mathematical models with varying degrees of sophistication and accuracy. Quantum electrodynamics forms the most complete of the models currently known, and produces the most accurate solutions when compared to physical measurements. However, physical accuracy usually comes at great computational expense, and indeed most models of light transport remain impractical for use in computer graphics today. Fortunately, many phenomena captured by sophisticated models, such as diffraction and interference, are usually subtle in the environments and wavelengths observed by human vision, and can be ignored with little loss in apparent realism. Where such effects are significant (e.g. interference in oil slicks, or diffraction in bird feathers), they can usually be injected locally in the form of specialized material models.
For these reasons, light transport simulation in computer graphics generally follows the simplest of these models, geometric optics. Geometric optics makes several simplifying assumptions that make computational treatment tractable: Light is incoherent and unpolarized, and travels at infinite speed along straight lines until an interaction with matter (i.e. scattering) occurs. This allows light to be described entirely by the balance, i.e. the inflow and outflow, of radiative energy at each interaction, as well as the transport of energy between interactions. We will sometimes informally refer to packets of radiative energy as “photons”; this is purely for illustrative purposes, as these “photons” are entirely unrelated to the much more complicated phenomenon of physical photons.

In the following sections, we will begin with a brief review of notation (Section 2.1), before describing the models we will use to represent discrete scatterers (i.e. surfaces) and probabilistic scatterers (i.e. participating media). We finish the preliminaries with a discussion of the quantities of interest in Section 2.3, and a brief review of scattering models in Section 2.4. With the preliminaries established, we introduce the energy balance equations of interest for this dissertation, and derive the transport and measurement equations that arise from them. We begin with discussing the balance equations separately for the case of discrete (Section 2.5) and probabilistic (Section 2.6) scatterers, before presenting the combined case (Section 2.7). These equations are recursive and do not readily lend themselves for advanced numerical simulation, and we therefore transform them first into a Cartesian reformulation (Section 2.8), before deriving the path integral and measurement equation we will use throughout the rest of the dissertation (Section 2.9).
2.1 Notation

For the rest of this dissertation, we will refer to vectors by bold letters, and use Roman script for scalars. Positions in space will usually use the Latin alphabet (e.g. $x, y, z,$ etc.), while we prefer the Greek letters $\omega$ and occasionally $\psi$ for unit directions. A position-direction pair $(x, \omega)$ defines a ray, i.e. a half-line beginning at $x$ and propagating in direction $\omega$. We will make use of the shorthand $x_t = x + t \omega$ to refer to points along rays; the dependence of $x_t$ on $x$ and $\omega$ is implicit here, and we only make use of this notation when $x$ and $\omega$ can be inferred non-ambiguously from context.

We will use the shorthand $\overrightarrow{xy}$ to refer to the unit direction from $x$ to $y$,

$$\overrightarrow{xy} = \frac{y - x}{\|y - x\|},$$

where we use $\|\cdot\|$ to denote the Euclidean (L2) norm. We will also use $x \cdot y$ to refer to the dot product of $x$ and $y$, and $|\cdot|$ to refer to the absolute value or the set cardinality, depending on context.

We will use calligraphic fonts to denote spaces and special sets; the only exception is the Euclidean space $\mathbb{R}^3$ for reasons of convention. Throughout this dissertation, we will introduce new spaces as needed; however, we define a few standard sets below that are used particularly often: The sphere $S^2$ containing all unit directions,

$$S^2 = \{\omega \mid \omega \in \mathbb{R}^3 \text{ and } \|\omega\| = 1\},$$

(2.2)

the ball $B(x, r)$ that contains all points within distance $r$ of the center $x$,

$$B(x, r) = \{y \mid y \in \mathbb{R}^3 \text{ and } \|y - x\| < r\},$$

(2.3)
and the cone of directions \( C(\omega, \theta) \) that contains all unit directions within an angle of \( \theta \) from \( \omega \),

\[
C(\omega, \theta) = \{ \omega' \mid \omega' \in S^2 \text{ and } \arccos \omega' \cdot \omega < \theta \}.
\]

(2.4)

While we avoid a detailed treatment of measures in this dissertation, we do need to integrate over spaces of varying dimensionality and structure, and will occasionally use measures to make this fact notationally explicit. Unless otherwise specified, we assume \( dx \) to be the usual Lebesgue measure on Euclidean space. Where we need to integrate over directions, we will use the solid angle measure\(^1\) \( d\Omega(\omega) \). Finally, we denote integrals over surfaces via the area measure \( dA(x) \) defined on 2D manifolds, to differentiate them from the volume measure \( dV(x) \) defined over proper subsets of 3D space. We refer to Jakob \([62]\) for an in-depth introduction to measures in graphics.

### 2.2 Scene Representation

In graphics, we typically differentiate between two different types of scatterers: surfaces and media. Surfaces are modeled as infinitesimally thin, two-dimensional manifolds that form the interface between media, whereas media fill the space between surfaces. For example (Fig. 2.1), a cup could be modeled with two media—the glass forming the interior and the air filling the exterior—and a surface that represents the interface between the two (and defines the shape of the cup).

We express this notion by defining the scene to be rendered over a space \( \Theta \subset \mathbb{R}^3 \), that can be decomposed\(^2\) into \( \Theta = A \sqcup V \), the set of surfaces (i.e. areas) \( A \) and the set of media (i.e. volumes) \( V \). The set of media is open, such that \( \partial V = A \) and \( V \cap A = \emptyset \).

---

1. The solid angle \( \Omega \), measured in steradians, is simply the 3D analog of the planar angle, measured in radians.
2. Here we use the disjoint union \( A \sqcup V \) instead of the union \( A \cup V \) of surfaces and media, so that they retain their separate topology in \( \Theta \).
2.2 SCENE REPRESENTATION

Figure 2.1: An illustrative example of the decomposition of a scene $\Theta$ into a set of surfaces $\mathcal{A}$ that form the interface between the set of media $\mathcal{V}$. Here, $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where $\mathcal{V}_1$ represents e.g. the air surrounding the cup, whereas $\mathcal{V}_2$ represents the glass material of the cup.

2.2.1 Surfaces

In modern rendering applications, there typically exist a myriad of ways to represent surfaces, ranging from analytic shapes such as spheres, conics or parametric splines \[121\] to discretized representations in the form of triangles, quadrilaterals and derived shapes \[22\]. In this dissertation, we are not overly concerned with the concrete representation used to model surfaces, as the work we present is largely orthogonal to the actual surface description in use.

For the rest of this discussion, we only require that each surface have the following set of properties: First, we assume we have access to a surface normal $N(x)$ which is a direction locally orthogonal to the surface\(^3\) at $x$ (see Fig. 2.1). Second, we assume we have access to a ray-tracing function $d(x, \omega)$ that returns the distance to the closest surface to $x$ along $\omega$,

$$d(x, \omega) = \min\{t \mid t > 0 \text{ and } x_t \in \mathcal{A}\}. \tag{2.5}$$

\(^3\) For representations with discontinuities (such as triangle meshes), the normal may not be uniquely defined at all points. However, as long as the set of such ambiguous points has measure zero (which is commonly the case), this is of no concern to the rendering algorithm.
We will use $x_A = x + d(x, \omega) \omega$ to refer to the closest surface point along a given direction, where again the dependence on $x$ and $\omega$ is made implicit.

### 2.2.2 Participating Media

While models that exclusively use surface representations have been used successfully, they are inherently limited. A large class of materials in the real world—such as clouds, smoke, or even human skin, for example—are neither perfectly opaque nor perfectly transparent. These materials form the class of *participating* media, i.e. media that actively partake in scattering. In reality, these media are themselves composed of an enormous number of microscopic scattering particles (that is, surfaces); however, explicitly modeling these scatterers would be wholly impractical due to their sheer number. Additionally, these particles are usually so small that diffraction effects make up a significant component of the light transport local to the medium, which would invalidate the geometric optics assumption.

A very useful approximation is to instead assume that—due to their microscopic size—the exact location of the particles is not important, and only their bulk effect on light has to be modeled. This permits a probabilistic treatment of participating media, where we only model the *likelihood* of encountering a scatterer along a particular ray through a medium. In a sense, photons travel undisturbed through the participating medium until they stochastically interact with it, at which point a concrete scatterer “manifests” directly in front of the photon.

The most basic property of a participating medium modeled this way is its *extinction coefficient* $\sigma(x) = \eta(x) \cdot \sigma$. It is characterized by the average cross-sectional area $\sigma$ of the scattering particles, as well as the number density $\eta(x)$, i.e. the average number of particles per unit volume.

To derive interaction probabilities from the extinction coefficient, we first need to define a model of how photons move through the medium. The prevalent model
in radiative transfer makes two fundamental assumptions: First, that a photon at location $x$ moving forward by an infinitesimal step of length $d t$ encounters a scattering particle with probability $\sigma(x) \cdot dt$; second, that the probability of encountering a scatterer at each infinitesimal step does not depend on any prior step \[23\]. Under this model, collisions with the medium form a *Poisson process*, and we can write the corresponding collision density:

$$p(x, x_t) = \sigma(x_t)e^{-\tau(x, x_t)} \quad \text{with} \quad \tau(x, x_t) = \int_0^t \sigma(x_s) \, ds,$$  \hspace{1cm} (2.6)

where $\tau$ is the so-called optical depth. The density $p$ is the *free-flight PDF*: It is the probability density that a photon emitted at point $x$ in direction $\omega$ will collide and interact with a scatterer at location $x_t$. It is a fundamental property of the medium.

A related, and useful probability measure is the *transmittance*. It expresses the probability that a photon emitted at $x$ passes location $x_t$ without interacting with the medium at any point in between:

$$\text{Tr}(x, x_t) = 1 - \int_0^t p(x, x_s) \, ds = \int_t^\infty p(x, x_s) \, ds = e^{-\tau(x, x_t)}. \hspace{1cm} (2.7)$$

This relationship can also be stated in reverse, and we can write $p$ in terms of $\text{Tr}$ as its derivative; the rate of decrease in transmittance is precisely the collision density:

$$p(x, x_t) = -\frac{d}{dt} \text{Tr}(x, x_t). \hspace{1cm} (2.8)$$

When $x$ and $\omega$ are unambiguous in context, we may occasionally use $\text{Tr}(t)$ and $p(t)$ to refer to the functions above.

Assuming independence of collisions makes for an appealing model, as it gives rise to efficient transport models and leads to an exponential transmittance with convenient properties. However, it fails to model participating media in which the
Figure 2.2: Radiance is the measurement of the amount of radiative energy, or “photons”, arriving from a small cone of directions of solid angle $d\Omega$ centered on direction $\omega$ that pass through a small surface patch of area $dA$ at location $x$ oriented orthogonally to $\omega$. Intuitively, it counts only photons that pass the entry aperture $d\Omega(\omega)$ in the illustration and subsequently hit area $dA(x)$.

locations of the scatterers are correlated. We lift this restriction and introduce a more general model in Chapter 4.

2.3 Radiometry

Up until this point, we have discussed light transport in terms of abstract concepts such as “light” or “photons”. It is therefore useful to first precisely state the quantity we wish to simulate, before discussing the equations that govern it.

Radiometry is the field of measuring radiation (including light), and it formalizes all measurable quantities of interest to us. A full review of radiometry is out of scope for this dissertation, and we refer readers to Veach [132] and Pharr et al. [111] for a complete discussion. For geometric optics, the main quantity of interest is radiance, and from this point forward, all equations will be in terms of this quantity.

Intuitively speaking, the radiance $L(x, \omega)$ describes the amount of illumination arriving from direction $\omega$ at point $x$. More precisely, radiance specifies the energy flow (measured in watts) passing through a (hypothetical) differential surface patch of area $dA$ oriented orthogonally to $\omega$ and centered on $x$. Only energy arriving from
Table 2.1: List of important symbols in this chapter, and their corresponding units.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_i, L_o, L_e, L_s)</td>
<td>(W \cdot \text{sr}^{-1} \cdot \text{m}^{-2})</td>
<td>Incident, exitant, emitted and scattered radiance</td>
</tr>
<tr>
<td>(\rho)</td>
<td>(\text{sr}^{-1})</td>
<td>Scattering function</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>(\text{m}^{-1})</td>
<td>Extinction coefficient</td>
</tr>
<tr>
<td>(p)</td>
<td>(\text{m}^{-1})</td>
<td>Free-flight PDF</td>
</tr>
<tr>
<td>(\text{Tr})</td>
<td>1</td>
<td>Transmittance</td>
</tr>
<tr>
<td>(V)</td>
<td>1</td>
<td>Visibility function</td>
</tr>
<tr>
<td>(G)</td>
<td>(\text{m}^{-2})</td>
<td>Geometry term</td>
</tr>
<tr>
<td>(dA)</td>
<td>(\text{m}^2)</td>
<td>Area measure</td>
</tr>
<tr>
<td>(dV)</td>
<td>(\text{m}^3)</td>
<td>Volume measure</td>
</tr>
<tr>
<td>(d\Omega)</td>
<td>(\text{sr})</td>
<td>Solid angle measure</td>
</tr>
</tbody>
</table>

To avoid ambiguity, we will further separate radiance into *incident radiance* and *exitant radiance*. Incident radiance \(L_i(x, \omega)\) specifies the amount of radiance arriving from direction \(\omega\) without having interacted (scattered) at \(x\). Exitant radiance \(L_o(x, \omega)\) specifies the radiance leaving in direction \(\omega\) immediately after having scattered at \(x\). We can relate the two via a balance of energy that arrives and leaves at equal rates, which will ultimately give rise to the light transport model we use throughout the rest of the dissertation.

It is worth pointing out that \(L_i(x, \omega)\) specifies radiance incident on a hypothetical surface patch orthogonal to \(\omega\). When dealing with surfaces however, we are interested in the radiance received by the actual surface, which is oriented orthogonal to \(N(x)\) instead. In this case, we can convert to radiance incident on the actual surface via the *foreshortening factor* \(|\text{cos} \theta| = |\omega \cdot N(x)|\), also sometimes referred to as Lambert’s cosine law [90].

Radiometry assigns physical units to fundamental quantities such as radiance. This allows for surface-level verification of the balance equations we present later.
in this chapter: Any equation expressing a balance of radiance must itself return radiance, and any change in units would point to a problem. For this purpose, we list the most important quantities of this chapter and their units in Table 2.1.

### 2.4 Scattering Models

Quintessential to the balance of incident and exitant radiance is the model that describes scattering at a point. We express scattering via a *scattering distribution function* $\rho(x, \omega_i, \omega_o)$ that describes how radiance arriving at $x$ from $\omega_i$ is redistributed into outgoing directions $\omega_o$. Because the scattering function is a distribution, it must obey certain properties: First, it must be nonnegative ($\rho \geq 0$); radiance cannot be “unradiated”. Second, it must observe energy conservation: Excluding self-emission, a scatterer cannot radiate more energy than it receives, thus

$$\int_{S^2} \rho(x, \omega_i, \omega_o) \, d\Omega(\omega_o) \leq 1 \quad \text{in media, and} \quad \int_{S^2} \rho(x, \omega_i, \omega_o) |\cos \theta| \, d\Omega(\omega_o) \leq 1$$

(2.9)

on surfaces. The scattering function is allowed to integrate to less than one to account for energy that is absorbed at $x$ and lost to further transport.

Although not mathematically required, we also assume that the scattering function obeys *reciprocity*, such that it behaves identically when incident and exitant direction are swapped:

$$\rho(x, \omega_i, \omega_o) = \rho(x, \omega_o, \omega_i).$$

(2.10)

Reciprocity not only makes the design of light transport algorithms more convenient by allowing transport from emitters to sensors to behave identically to the reverse
Figure 2.3: We show three different types of scattering on surfaces and media: Diffuse (a), where the distribution (blue) does not depend on incident direction (red); glossy (b), when there is some dependence; and specular (c), when radiance is scattered into a discrete set of exitant directions.

transport, but is also a property generally observed in nature. Veach introduces a more precise reciprocity principle in the presence of refractive materials, but the simpler form in Eq. (2.10) will suffice for our purposes.

Deviating somewhat from existing convention, we use the same function to refer to scattering in media and on surfaces. Traditionally, scattering in media would be described by the product of the albedo $\alpha$ and a normalized directional distribution, the phase function $\rho_m$. Surface scattering would instead be described by the bidirec-

---

4 Barring other effects, such as fluorescence.
tional scattering function (BSDF) $\rho_s$ that is unnormalized and implicitly incorporates both albedo and directional distribution. To add to the confusion, phase function and BRDF use conflicting conventions, where the direction $\omega_i$ is assumed to point toward the location of scattering for phase functions, but points towards the direction of incoming light for BSDFs. We opt to describe media- and surface scattering using the same interface $\rho$ that uses the BRDF convention; for media, we can assume the necessary conversions to happen “under the hood”.

Similar to surface representations, there exist a vast number of scattering models in use today. Our algorithms generally work independently of the type of scattering model(s) used, and we refer to Pharr et al. [110] for an overview of existing models. However, it is sometimes useful to refer to the character of scattering, and we will sometimes characterize scattering as “diffuse” when its distribution is independent of $\omega_i$; “glossy” when there is some dependence between $\omega_i$ and $\omega_o$; and “specular” when radiance is scattered into only a discrete set of directions. We illustrate these three different types of scattering for surfaces and media in Fig. 2.3.

### 2.5 Balance on Surfaces

With the fundamental properties of our scene established, we can now begin to derive the light transport model we will use for the rest of this dissertation. We begin our derivation with the simple case of pure surface scattering, where media are assumed not to participate in transport. This means that within media, radiance along rays is fully conserved, i.e. the directional derivative $(\omega \cdot \nabla) L(x, \omega)$ of radiance is zero:

$$ (\omega \cdot \nabla) L(x, \omega) = 0. \quad (2.11) $$
Radiance constancy along rays allows us to directly relate the radiance \( L_i(x, \omega_i) \) incident on a surface at \( x \) to the radiance leaving the closest surface via \emph{ray-tracing}:

\[
L_i(x, \omega_i) = L_o(x_{A_r}, -\omega_i). \tag{2.12}
\]

In turn, the radiance \( L_o(x, \omega_o) \) leaving a surface after scattering is comprised of two components: The \emph{emitted radiance} \( L_e(x, \omega_o) \) directly emitted at \( x \), and the \emph{scattered radiance} \( L_s(x, \omega_o) \), which is the result of radiance incident on \( x \) from other directions and scattered toward \( \omega_o \):

\[
L_o(x, \omega_o) = L_e(x, \omega_o) + L_s(x, \omega_o). \tag{2.13}
\]

The scattered radiance is comprised of an integral that sums how much light arrives at \( x \) (expressed via \( L_i \) and the foreshortening factor), and is scattered into \( \omega_o \) (expressed via the scattering function):

\[
L_s(x, \omega_o) = \int_{S^2} \rho(x, \omega_i, \omega_o)L_i(x, \omega_i)|\cos \theta_i| \, d\Omega(\omega_i). \tag{2.14}
\]

This system of balances is a full accounting of all surface transport: Energy is injected into the scene through emission \( (L_e) \), and redistributed through scattering \( (L_s) \) and transport \( (\text{Eq. (2.12)}) \). The steady state solution of these equations uniquely describes the radiance distribution in the scene.

2.6 Balance in Participating Media

Analogous to before, we now assume a scene composed exclusively of a medium and containing no surfaces. The balance equations in this case are remarkably similar to the pure surface case: Radiance is still emitted and scattered by “surfaces”,

except now contact with these surfaces is probabilistic. Using the definition from Section 2.2.2, the probability of colliding with a scatterer after travelling an infinitesimal distance \( dt \) is \( \sigma(x) \cdot dt \); this allows us to directly write the change in radiance along a ray:

\[
(\mathbf{w} \cdot \nabla)L(x, \mathbf{w}) = \sigma(x) \cdot (-L(x, \mathbf{w}) + L_e(x, \mathbf{w}) + L_s(x, \mathbf{w})).
\] (2.15)

This equation is called the equation of radiative transfer (RTE) [23]. It is comprised of three terms: The first term removes radiance that interacts with the scatterer. In a sense, in the surface case (Section 2.5) all radiance incident on a surface was removed and replaced with the exitant radiance (i.e. emission and scattering). In the case of a participating medium however, only the fraction \( \sigma(x) \) that actually collides with a scatterer undergoes this change, leading to the inclusion of the first term. The second and third term account for the radiance added due to self-emission at the particle and due to scattering from other directions.

Rewriting this equation explicitly in terms of exitant radiance, we now obtain the balance equations for \( L_o \) and \( L_s \), assuming that a collision occurred:

\[
L_o(x, \mathbf{w}_o) = L_e(x, \mathbf{w}_o) + L_s(x, \mathbf{w}_o)
\] (2.16)

\[
L_s(x, \mathbf{w}_o) = \int_{S^2} \rho(x, \mathbf{w}_i, \mathbf{w}_o)L_i(x, \mathbf{w}_i) d\Omega(\mathbf{w}_i).
\] (2.17)

Compared to the surface case (Eq. (2.13) and Eq. (2.14)), these equations are identical except for a missing foreshortening factor; the projection of the scattering surface is already incorporated into the particle’s cross-sectional area (and hence \( \sigma \), accounted for shortly) and makes the cosine unnecessary.

The incident radiance, on the other hand, differs significantly from the surface case. This is because there is no longer just a single point responsible for the radiance incident at \( x \), but instead all locations \( x_t \) along direction \( \mathbf{w} \) can potentially contribute.
At each such location, light is emitted or scattered with probability $\sigma(x_t) \cdot dt$. Given that an interaction occurred, $L_o(x_t, -\omega)$ is the radiance contributed to the ray, of which the fraction $\text{Tr}(x, x_t)$ reaches $x$. Summing over the product of these terms over all points $x_t$ along the ray then forms the incident radiance:

$$L_i(x, \omega_i) = \int_0^\infty \text{Tr}(x, x_t) L_o(x_t, -\omega_i) \sigma(x_t) dt.$$  \hfill (2.18)

These are the full balance equations describing transport in a scene consisting exclusively of participating media.

### 2.7 Combined Balance

With the balance equations for surfaces and media established in isolation, we are now ready to derive the combined balance. The similarity of $L_o$ and $L_s$ on surfaces and in media allows us to immediately write down their analog for scenes containing both:

$$L_o(x, \omega_i) = L_e(x, \omega_o) + L_s(x, \omega_o)$$  \hfill (2.19)

$$L_s(x, \omega_o) = \int_{S^2} \rho(x, \omega_i, \omega_o) L_i(x, \omega_i) D(x, \omega_i) d\Omega(\omega_i),$$  \hfill (2.20)

where the foreshortening term $D$

$$D(x, \omega_i) = \begin{cases} 1 & \text{if } x \in V \\ |N(x) \cdot \omega_i| & \text{if } x \in A \end{cases}$$  \hfill (2.21)

simply expresses the differences in geometry between the surface and medium case. These equations reduce to Eq. (2.14) and Eq. (2.17) respectively, depending on whether $x$ lies on a surface or not.
The final piece is the incident radiance, which is the sum of radiance incident from points in the medium (Eq. (2.18)) and the closest surface point (Eq. (2.12)):

\[ L_{i}(x, \omega_{i}) = \int_{0}^{d} \text{Tr}(x, x_{t}) L_{o}(x_{t}, -\omega_{i}) \sigma(x_{t}) \, dt + \text{Tr}(x, x_{A}) L_{o}(x_{A}, -\omega_{i}). \] (2.22)

The only alterations are a change of integral bounds so that radiance is only gathered from points in the medium that are in front of the nearest surface, and an additional \( \text{Tr} \) term to only account for uncollided surface radiance, i.e. radiance leaving \( x_{A} \) without being lost to the medium. These are the complete balance equations for arbitrary scenes, and the rest of our theory will be based on this model of transport.

### 2.8 Cartesian formulation

The balance equations we derived in the previous sections are complete in that they describe all transport of interest, but they are not yet in a form that can conveniently express most modern rendering algorithms: The presence of the ray-tracing operator in Eq. (2.22) implicitly prescribes the direction in which transport needs to be simulated, and the recursive nature of these equations make them difficult to manipulate. In the following sections, we will therefore transform these equations first into a Cartesian formulation that integrates over points instead of directions, which removes the ray-tracing operator, before finally “unrolling” the recursion into a sum of integrals in Section 2.9.

We begin by inserting \( L_{i} \) into \( L_{s} \); the purpose of this is to obtain a combined integral over directions and distances. We then perform a change of variables from
direction/distance measure into the more straightforward measure over Cartesian coordinates (see Fig. 2.4). Expanding Eq. (2.20) yields

\[ L_s(x, \omega_o) = \int_{S^2} \rho(x, \omega_o, \omega_i) \text{Tr}(x, x_A) L_o(x_A, -\omega_i) \Sigma(x) D(x) d\Omega(\omega_i) \]

\[ + \int_{S^2 \times [0,d]} \rho(x, \omega_o, \omega_i) \text{Tr}(x, x_1) L_o(x_1, -\omega_i) \Sigma(x) D(x) dt d\Omega(\omega_i). \quad (2.23) \]

where we have used the scattering term \( \Sigma \)

\[ \Sigma(x) = \begin{cases} 
\sigma(x) & \text{if } x \in \mathcal{V} \\
1 & \text{if } x \in \mathcal{A}
\end{cases} \quad (2.24) \]

analogous to the foreshortening term \( D \) (Eq. (2.21)).

We immediately notice that both terms in the above equation use essentially the same integrand; the only difference is which points (and which measure) they use for integration. Using a specific change of variables, we will be able to unify both of
these integrals into one: We change from integrating over directions and distances to integration over points on surfaces and volumes directly. This incurs the Jacobians

\[
\begin{align*}
\frac{d\Omega(\omega)}{d\Omega(A)} &= \frac{V(x, x_A)}{\|x - x_A\|^2} \left( \frac{D(x_A, \omega)}{2} \right) \, d\Lambda(x_A) \\
\frac{dt \, d\Omega(\omega)}{d\Omega(1)} &= \frac{V(x, x_1)}{\|x - x_1\|^2} \left( \frac{D(x_1, \omega)}{2} \right) \, dV(x_1).
\end{align*}
\]

These Jacobians account for the changes in measure caused by the different coordinate systems: Points distributed uniformly along rays in direction/distance coordinates have a density fall-off in Cartesian coordinates inversely proportional to the square distance to the ray origin \(x\). Analogous to Lambert’s cosine law, the density of ray-traced points distributed uniformly over directions decreases in area measure as surfaces tilt away from the ray direction, incurring an additional foreshortening factor \(D\). Finally, where it was easy to reason about “closest points” and “in front of” in direction/distance space, the Cartesian formulation will integrate over all points in the scene. To account for this, we make use of the visibility term

\[
V(x, x_t) = \begin{cases} 
1 & \text{if } t \leq d(x, \omega) \\
0 & \text{otherwise}
\end{cases}
\]

which expresses these concepts in terms of an indicator function that equals 1 if the point \(x_t\) is in front of or equal to the closest surface point.

To simplify notation, we will gather all common terms into the geometry term \(G\):

\[
G(x, y) = \frac{V(x, y) \left( \frac{D(x, y)D(y, \bar{y})}{\|x - y\|^2} \right)}{\left( \left( \frac{D(x, \omega)}{2} \right) \right) \, d\Lambda(x)}.
\]

Using these mathematical tools, we can now rewrite Eq. (2.23) using a change of variables:

\[
L_s(y, \bar{y}) = \int \rho(y, \bar{y}, \bar{z}, \bar{y}) \left( \text{Tr}(y, z) L_\text{o}(z, \bar{y}) \Sigma(y) G(y, z) \right) \, d\Lambda(z) \\
+ \int \rho(y, \bar{y}, \bar{z}, \bar{y}) \left( \text{Tr}(y, z) L_\text{o}(z, \bar{y}) \Sigma(y) G(y, z) \right) \, dV(z).
\]
This is sometimes referred to as the three-point form: It describes the radiance leaving point \( y \) toward point \( x \), summed over all points \( z \) that contribute to the radiance at \( y \) (see Fig. 2.4).

To unify the two integrals over surfaces and media, we equip the scene space \( \Theta = A \sqcup V \) with an appropriate measure \( d\mu(x) \) such that

\[
\int_{\Theta} f(x) \, d\mu(x) = \int_{V} f(x) \, dV(x) + \int_{A} f(x) \, dA(x). \tag{2.30}
\]

With this, we obtain the final form of the Cartesian formulation:

\[
L_s(y, \bar{y} \bar{x}) = \int_{\Theta} \rho(y, \bar{y} \bar{z}, \bar{y} \bar{x}) \mathcal{T}r(y, z) L_o(z, \bar{y} \bar{z}) \Sigma(y) G(y, z) \, d\mu(z). \tag{2.31}
\]

The above equation, restricted to the surface case only, is equivalent to the surface area formulation of the rendering equation introduced in the seminal work of Kajiya \[78\] , which forms the foundation of all light transport models in use in graphics today.

### 2.9 The Path Integral and the Measurement Equation

The Cartesian formulation in Eq. (2.31) unifies the surface- and media integrals and operates in the much more convenient space of \( \Theta \). However, this equation is still recursive, which complicates the design of algorithms used to solve it. In this section, we manipulate the Cartesian form to finally derive the non-recursive path integral formulation introduced by Veach \[131\], which will form the basis for the rendering algorithms we discuss in all subsequent chapters.

The key observation in the path integral formulation is that \( L_o \) in Eq. (2.31) is the sum of two terms: \( L_e \), which can be evaluated directly, and \( L_s \), which requires recursive evaluation. Because of the linearity of integration, the integral of the sum
can be rearranged to obtain an infinite sum of integrals instead: Each term in this sum is a finite nesting of integrals over $L_s$, with the last term being $L_e$.

Veach [131] formalizes this notion in terms of finite-length paths $\vec{x} = x_0 x_1 \ldots x_k$ of length $k$. Each path consists of $k + 1$ vertices $x_i$ and $k$ segments. Each integral in the infinite sum mentioned above operates on paths of a fixed length. The domain of each integral is the space of paths of length $k$:

$$P_k = \Theta \times \cdots \times \Theta.$$

The core of the integrand is formed by the path throughput function $g(\vec{x})$, which gathers all terms of recursing Eq. (2.31) $k - 1$ times (Fig. 2.5), excluding the final $L_e$ term:

$$g(\vec{x}) = \Sigma(x_0) \left[ \prod_{i=1}^{k-1} \rho(x_i, x_{i-1}, x_{i+1}) \Sigma(x_i) \right] \left[ \prod_{i=0}^{k-1} \text{Tr}(x_i, x_{i+1}) G(x_i, x_{i+1}) \right] \Sigma(x_k).$$

Given a path $\vec{x}$, this function simply expresses the intuitive notion of how much of the light entering the path at $x_0$ arrives at $x_k$. Using the path throughput, we can readily express the radiance arriving at $x_k$ by multiplying with the emitted radiance, $L_e(x_0, x_0 x_1) g(\vec{x})$. 

Figure 2.5: A path of length 3 and the corresponding terms of the measurement contribution along the path.
2.9.1 Measurement Equation

To complete the full transport model, we finally introduce the notion of a measurement. Real sensors, such as pixel elements in cameras or receptor cells in human vision, do not take point measurements of radiance, but rather receive radiance over some finite area and solid angle. We can make concrete this notion via the importance $W_c(x, \omega)$ that describes the sensitivity of the sensor for any given point and direction.

A measurement of radiance taken by the sensor can be expressed as an integral over all possible paths of all lengths that connect an emitter to the sensor. This gives rise to the measurement equation

$$I = \int_{\mathcal{P}} f(x) \, d\mu(x),$$

(2.34)

where $f$ is the measurement contribution function

$$f(x) = L_e(x_0, \bar{x}_0 \bar{x}_1) g(x) W_e(x_k, \bar{x}_k \bar{x}_{k-1})$$

(2.35)

and $\mathcal{P}$ is the path space, i.e. the space of all paths of all lengths:

$$\mathcal{P} = \bigcup_{k=1}^{p^k}$$

(2.36)

This is the final and complete form of the light transport model on which all modern light transport algorithms are based. It is a powerful framework that can represent a diverse set of image formation models, and in which many different designs of rendering algorithms can be expressed.
2.9.2 A Note on Notation

To be fully precise, we should refer to measurements as $I_j$ over corresponding contribution $f_j$ to disambiguate the pixel $j$ at which the measurement is taken. However, to keep notation concise, we will drop the subscript unless we explicitly need to refer to multiple measurements.

Although paths are explicitly expressed in terms of points $x_i$, it is sometimes more natural to write the path contribution in terms of directions and distances instead, and we will use the following convention to convert between the two:

$$\omega_i = x_{i-1} x_i \quad \text{and} \quad t_i = \|x_i - x_{i-1}\| \quad \text{such that} \quad x_i = x_{i-1} + t_i \omega_i. \quad (2.37)$$

Even so, equations involving the full measurement contribution can sometimes become cumbersome, and we use the shorthands

$$\rho(x_i) = \rho(x_i, x_{i-1} x_{i+1}) \quad (2.38)$$
$$L_e(x_0) = L_e(x_0, x_0 x_1) \quad (2.39)$$
$$W_e(x_k) = W_e(x_k, x_k x_{k-1}) \quad (2.40)$$

to keep notation concise.

Finally, where the length of a path is important and cannot be inferred from context, we will use $\overline{x}_k$ to explicitly denote a length-$k$ path. Care must be taken as not to confuse $\overline{x}_k$ with the $k$-th vertex $x_k$. 
The measurement equation introduced in the previous chapter establishes a well-defined mathematical framework for representing transport of light. In this chapter, we now turn to solving this equation.

Solving difficult integral equations poses a major challenge in many fields of science. Save for special cases, analytic solutions are rarely available, and what remains are numerical methods that are able to compute approximate solutions to integrals. Although a plethora of numerical integration methods exist, their efficiency generally depends on the specific form of the integral, and only few have been applied successfully to the measurement equation. Early approaches include finite element methods such as Radiosity [49], but beginning with Cook et al. [28] and Kajiya [78], Monte Carlo (MC) based methods quickly became the de facto solution technique to the measurement equation.

In this chapter, we will review the preliminaries and established rendering algorithms in use today on which our novel algorithms (Chapter 5–Chapter 9) are based.
3.1 Basics of Monte Carlo Integration

Monte Carlo integration is a method for approximating the value of any integral to within arbitrarily small expected error. Given an integral of interest

\[ I = \int_\Theta f(x) \, dx, \quad (3.1) \]

Monte Carlo integration constructs a random variable \( Y \) such that the expected value of \( Y \) matches the desired integral, i.e. \( E[Y] = I \). Given such a primary estimator, it is trivial to construct a secondary estimator that approximates \( I \) to within any desired expected error

\[ I \approx I^{(N)} = \frac{1}{N} \sum_{i=1}^{N} y_i \quad (3.2) \]

given \( N \) independent realizations \( y_1, \ldots, y_N \) of \( Y \). Due to the law of large numbers, \( I^{(N)} \) converges to \( I \) as \( N \to \infty \). However, for any finite \( N \), the estimate will contain some error; this manifests as random deviations—noise—from the true solution.

3.1.1 Variance

We can help characterize the magnitude of this noise by the variance \( \mathbb{V}[I^{(N)}] \) of the Monte Carlo estimator. We know from the central limit theorem that as \( N \) increases, the distribution of the error tends to a zero-mean normal distribution with standard deviation \( \sqrt{\mathbb{V}[I^{(N)}]} \). If the variance is known and finite, then for any large \( N \) we can reasonably bound \( I^{(N)} \) to be within some distance of the true value with known probability.
If the primary estimator has (finite) variance $\mathbb{V}[Y]$, then the secondary estimator has variance

$$
\mathbb{V}[I(N)] = \frac{1}{N} \mathbb{V}[Y].
$$

As expected, variance tends to zero in the limit and the error vanishes. Although we rarely know $\mathbb{V}[Y]$ a priori, we can make statements about the asymptotics of $\sqrt{\mathbb{V}[I(N)]}$; in particular, in order to reduce the expected integration error by half, we need to increase $N$ by a factor of 4.

This means that decreasing the expected error by increasing $N$ is always possible regardless of the integration problem. However, Monte Carlo suffers from a problem of diminishing returns, wherein the superlinear growth in $N$ means that arbitrarily low error bounds are not feasible to achieve this way in practice. Instead, we must turn to reducing the variance of the primary estimator $Y$.

### 3.1.2 Primary Estimators

In theory, any random variable $Y$ with $\mathbb{E}[Y] = I$ is a valid Monte Carlo estimator for $I$. In practice however, the overwhelming majority of estimators take the form

$$
Y = \frac{f(X)}{p_X(X)},
$$

where $p_X(X)$ is the probability distribution function (PDF) of random variable $X$. As long as $p_X(X) > 0$ where $f(X) \neq 0$, this estimator has the desired expected value:

$$
\mathbb{E}[Y] = \int_{\Theta} \left( \frac{f(X)}{p_X(X)} \right) p_X(X) \, dX = \int_{\Omega} f(X) \, dX = I.
$$

Remarkably, the secondary estimator may converge to the expected value even if the variance of the primary estimator is infinite. However, the asymptotic rate of convergence is slower than in the finite case. The exact convergence rate in the case of infinite variance depends on the estimator itself, and may not be straightforward to derive.
Although Eq. (3.4) represents the most common Monte Carlo estimator in use in graphics today, others are certainly possible; control variates [94] are another example of MC estimators that have recently gained popularity in graphics [119]. Indeed, later in this dissertation (Chapter 4) we will encounter another form of Eq. (3.4) where \( p_X(X) \) is not known and itself is estimated.

The variance of the estimator in Eq. (3.4) directly depends on the variability of \( f(X)/p_X(X) \). To achieve low variance, it is thus imperative to choose \( p_X \) to be as proportional to \( f \) as possible; this is called importance sampling. Achieving this goal for the measurement equation has been the subject of major research efforts over the past 30 years, some of which we will review in Section 3.2.

### 3.1.3 Bias and Consistency

So far, we have discussed estimators whose expected value matches the desired solution. We can express this property in terms of the bias:

\[
\mathbb{B}[Y] = \mathbb{E}[Y] - I. \tag{3.6}
\]

The estimators up until now satisfy \( \mathbb{B}[Y] = 0 \), and are called unbiased estimators. However, there also exist a number of useful biased estimators (indeed, Chapter 5, Chapter 8 and Chapter 9 will introduce three new ones). Bias does not mean arbitrarily low error cannot be reached; even if the expected value does not match the integral, a biased estimator can sometimes still be consistent, where the bias vanishes as the sample size tends to infinity.

To give an intuitive example, consider the task of estimating the expected number showing on a fair dice. Given a primary estimator \( X \) that takes on uniformly
distributed (integer) values between 1 and 6, consider the following secondary estimators:

\[
A = \frac{1}{N} \sum_{k=1}^{N} x_k \quad B = \frac{1}{N} + \frac{1}{N} \sum_{k=1}^{N} x_k \quad C = 1 + \sum_{k=1}^{N} x_k \quad D = x_1
\]  

Estimator A is the standard MC secondary estimator we have seen earlier, and is both unbiased and consistent. B is a biased estimator, but the bias (here, \( B[B] = 1/N \)) vanishes as \( N \to \infty \), making it consistent. C is both biased and inconsistent; the bias does not diminish with \( N \), and the estimator will not approach the true solution. Finally, D is an unbiased but inconsistent estimator: Even though it clearly has the correct expected value, it will not converge to the true mean as \( N \) increases (indeed, it stays constant).

All of these four types of estimators appear in graphics. Although unbiasedness can be desirable e.g. if additional statistical analysis (such as denoising or error estimation) is performed on the estimator, bias and variance are ultimately only statistical properties and do not speak to the usefulness of a particular estimator. Instead, we are usually interested in metrics of the estimation error, such as the mean squared error (MSE):

\[
\text{MSE}(N) = E \left[ (I^{(N)} - I)^2 \right] = B^2 \left[ I^{(N)} \right] + V \left[ I^{(N)} \right],
\]

which incorporates both bias and variance.

Regardless of bias or even consistency, most practical applications are simply interested in the estimator with the lowest amount of error for a given finite \( N \) (the computational budget); the behavior of the estimator for \( N \to \infty \) is less of a concern.
3.1.4 Importance Sampling

For our purposes, importance sampling is an umbrella term referring to methods that generate samples from distributions tailored to sample a specific integrand $f(x)$ with as little variance as possible. While importance sampling high-dimensional integrands can be a formidable challenge in general, the functions we consider in light transport consist of the product of many lower-dimensional terms. Luckily, there exist a number of recipes that can readily importance sample simple and low-dimensional functions, sometimes perfectly, and we will review some of these recipes in the following sections. More sophisticated sampling strategies can then be obtained by the composition of lower-dimensional sampling schemes.

3.1.4.1 The Inversion Method

The inversion method, also sometimes called inverse transform sampling, is a general technique for deriving perfect importance sampling schemes. Given a target distribution $p(x)$ (here, in 1D) that we wish to sample, the inversion method first computes the cumulative distribution function (CDF) and its inverse (the ICDF) of this PDF:

$$\text{cdf}(x) = \int_{0}^{x} p(y) \, dy \quad \text{and} \quad \text{icdf}(\xi) = \text{cdf}^{-1}(\xi). \quad (3.9)$$

If both of these functions exist and can be derived in closed form, then $\text{icdf}(\xi)$ (with $\xi$ a random number in $[0, 1]$) produces random samples distributed exactly with $p(x)$. However, doing so requires computing both the integral of the desired distribution and its inverse in closed form, and either step can fail depending on the distribution.

The inversion method can also be applied to multi-dimensional distributions. We refer to Pharr et al. [110] for details.
3.1.4.2 Resampled Importance Sampling

Where a desired distribution $\hat{p}(x)$ cannot be sampled directly using the inversion method, resampled importance sampling (RIS) \(^{128}\) can be used instead as a method for producing samples distributed approximately proportional to $\hat{p}(x)$. Given a candidate distribution $p(x)$ which can be sampled, RIS generates $M$ candidates $x_1, \ldots, x_M$ from $p(x)$ and randomly selects a sample $x_z$ to be used as the output sample. $x_z$ is selected with probability proportional to the selection weight

$$w(x) = \frac{\hat{p}(x)}{p(x)}.$$  \hspace{1cm} (3.10)

As $M \to \infty$, the distribution of $x_z$ converges to the desired distribution $\hat{p}(x)$. Notably, this procedure does not require $\hat{p}(x)$ to be normalized, which allows for great freedom in choosing the target distribution.

Samples produced by RIS must be used in a primary estimator that differs from the standard form in Eq. (3.4):

$$I_{\text{RIS}} = \frac{f(x_z)}{\hat{p}(x)} \frac{1}{M} \sum_{i=1}^{M} w(x_i).$$ \hspace{1cm} (3.11)

We investigate this estimator in depth in Chapter 8 and Chapter 9 and show how it can be used to build a new class of rendering algorithms.

3.1.4.3 Other Importance Sampling Techniques

A variety of other schemes exist that can sample general or specific integrands exactly or approximately.

For example, rejection sampling is a method for producing uniform samples over arbitrary sets that are difficult to sample with other methods. It operates by generating samples uniformly over a simpler bounding shape (e.g. a cuboid) that fully contains the set of interest, and keeps drawing new samples until one is obtained.
that lies in the set of interest. Interestingly, this procedure can be readily applied to sample any nD distribution by lifting it to (n + 1)D dimensions. For example, sampling a 1D distribution \( p(x) \) can be reformulated as uniformly sampling the 2D shape formed by the area under the curve of \( p(x) \). Assuming the bounds of \( p(x) \) are known and finite, we can then use rejection sampling to sample the area uniformly, and discard the \((n + 1)\)th dimension to obtain a sample distributed according to \( p(x) \).

Gibbs sampling \([43]\) is a related idea, in which the next sample is generated from the previous sample by regenerating only one of its dimensions, which is equivalent to moving along a line in the shape defined by the lifted distribution. Samples produced by this process are no longer independent, but form a Markov Chain. We will review the very related class of Markov Chain Monte Carlo rendering algorithms in Section 3.2.3.

### 3.1.5 Multiple Importance Sampling

Although a difficult integrand such as the measurement contribution cannot be sampled perfectly, we usually have a variety of sampling schemes available that each approximately sample the integrand, e.g. by sampling different subsets of its terms using one of the methods we reviewed in the previous subsections.

Given a variety of sampling schemes, it is not usually obvious which scheme should be used to obtain the lowest variance, and the answer may be different for each transport path. Instead of selecting a single sampling scheme, a more robust approach is to generate samples from all schemes, and use their weighted combination. Given \( k \) sampling schemes that produce samples distributed according to
\( p_1(x), \ldots, p_k(x) \), Veach et al. [134] propose the multiple importance sampling (MIS) estimator

\[
I_{\text{MIS}} = \sum_{i=1}^{k} m_i(x_i) \frac{f(x_i)}{p_i(x_i)}, \tag{3.12}
\]

where \( m_i(x) \) is the MIS weight of the \( i \)th sampling scheme. Any assignment of weights that satisfies \( \sum_{i=1}^{k} m_i(x) = 1 \) is a valid MIS heuristic. However, of special interest are weighting heuristics that provably reduce variance compared to using only a single estimator, or their unweighted average.

Veach et al. propose several MIS weighting strategies that are provably optimal\(^2\), assuming positive weights. Most commonly used is the balance heuristic

\[
m_i(x) = \frac{p_i(x)}{\sum_{j=1}^{k} p_j(x)} \tag{3.13}
\]

which preferentially selects sampling schemes with a relatively large PDF for a given sample. Occasionally useful is also the maximum heuristic

\[
m_i(x) = \begin{cases} 
1 & \text{if } i = \arg \max_j p_j(x) \\
0 & \text{else} 
\end{cases} \tag{3.14}
\]

that assigns non-zero weight only to the technique with the largest PDF for a given sample.

Variance can be even further reduced if we allow for negative MIS weights [82]. Such weighting schemes tend to be much more challenging to compute however, and we will restrict ourselves to the (comparatively simple) original heuristics of Veach et al. for this dissertation.

\( ^2 \) In the sense that no other heuristic can exhibit variance much lower than the proposed heuristics.
3.2 Monte Carlo Rendering Algorithms

The main goal of Monte Carlo rendering algorithms is to estimate the measurement equation with as little expected error as possible. In this context, the integrand is the measurement contribution function \( f(\mathbf{x}) \), and each sample \( X \) is comprised of a transport path \( \mathbf{x} \). The goal of a rendering algorithm is then to generate paths with distribution \( p_X(\mathbf{x}) \) as proportional to \( f(\mathbf{x}) \) as possible.

3.2.1 Light-, Path- and Bidirectional Path Tracing

In general, perfect importance sampling of \( f \) is impossible due to its complexity and high dimensionality. However, \( f \) itself is simply the product of (possibly many) low-dimensional functions that are only conditioned on neighboring vertices on the path (Eq. (2.33)), for which sampling routines are readily available.

It is therefore possible to generate paths whose distribution matches nearly all terms of the path contribution by starting at one end of the path, and generating all subsequent vertices by importance sampling subsequent terms of the path contribution.

For example, we could generate a path with the following procedure:

1. Generate \( x_0 \) by importance sampling the marginal \( L_e(x_0, \cdot) \)
2. Generate \( \omega_1 \) by importance sampling \( L_e(x_0, \omega_1) \) (with \( x_0 \) fixed)
3. Generate \( t_1 \) by importance sampling \( \mathcal{T}_e(x_0, x_0 + t_1 \cdot \omega_1) \) (with \( x_0 \) and \( \omega_1 \) fixed)
4. Generate \( \omega_2 \) by importance sampling \( \rho(x_1, \omega_1, \omega_2) \) (with \( x_1 \) and \( \omega_1 \) fixed)
5. Go to step 2 until desired path length is reached.

This procedure is called *light tracing* [1]; if we had started from the other end of the path instead (i.e. by sampling \( W_e \) in step 1 and 2), we would obtain *path tracing* [78].
We can also generate any method in between these two by their combination: We sample a length-\(l\) light path \(x_l\) via light tracing and a length-\(k\) sensor path \(y_k\) via path tracing to obtain the combined length-(\(l + k + 1\)) path \(x_0 \ldots x_l y_k \ldots y_0\).

All of these techniques importance sample the path contribution perfectly, save for one term: Light tracing does not importance sample the importance term \(W_e\); path tracing does not importance sample the emission term \(L_e\); and any combination of them does not importance sample the segment term \(\text{Tr}(x_l, y_k)G(x_l, y_k)\).

Sadly, any one of these terms can introduce arbitrarily high variance, and selecting the ideal estimator for a given scene is not straightforward. An alternative is to use all of the aforementioned estimators simultaneously and combine their contributions using multiple importance sampling. This method is called bidirectional path tracing [87, 88, 133] (BDPT). BDPT is among the most robust rendering algorithms, and exhibits variance comparable to the lowest variance of any of its sub-estimators at any given sample count. However, combining this many estimators comes at considerable computational expense, and when comparing variance at a fixed computational budget, BDPT can be significantly outperformed by less sophisticated algorithms when the additional robustness is not required.

### 3.2.2 Photon Mapping

The path sampling techniques reviewed in the previous section all suffer from the same flaw, which is that at least one term in the measurement contribution cannot be importance sampled and potentially introduces high variance. Photon mapping [44, 58, 74, 75] presents an interesting solution to this problem. Consider the bidirectional path construction technique discussed earlier: Subpaths \(\bar{x}\) and \(\bar{y}\) are constructed from an emitter and a sensor, respectively, and are joined to form the path \(\bar{z} = x_0 \ldots x_l y_k \ldots y_0\). The connecting segment between \(x_l\) and \(y_k\) is determined by the location of the subpath endpoints, and cannot be chosen (i.e. sampled) freely.
(Fig. 3.1 (a)). This is the source of the remaining unsampled terms in the measurement contribution, and thus the variance.

If $x_l$ and $y_k$ happen to be very close together, we could instead form the alternative path $x_0 \ldots x_{l-1} y_k \ldots y_0$, which omits one of the vertices and removes the unsampled segment between $x_l$ and $y_k$ entirely \cite{44, 58}. The only remaining unsampled term is the scattering function $\rho$ at $y_k$. There is also a change in length and direction when exchanging segment $x_{l-1} x_l$ with segment $x_{l-1} y_k$, but as long as $x_l$ and $y_k$ are close together, this would hopefully not introduce too much variance. The path is conditionally accepted using a predicate $K(x_l, y_k)$ that encapsulates the concept of “closeness” between $x_l$ and $y_k$, e.g. by thresholding their distance (see Fig. 3.1 (b)).

**A two-pass algorithm.** There are several hurdles that need to be cleared before such an algorithm can be made to work. First, the endpoints of two subpaths only rarely land close together, and the predicate $K$ would rarely be met. Photon mapping solves this problem by tracing many light subpaths at once, which increases the likelihood of being able to successfully connect any one of them with the sensor subpath. To reduce the computational cost of this approach, the same set of light subpaths is
reused for all sensor subpaths in order to amortize the computation over the whole rendering: In a first phase, a large number of light subpaths is generated up-front, and their endpoints stored in a spatial data structure. This allows rapidly finding nearby light paths in the subsequent second phase when generating sensor paths. We can additionally exploit that any prefix $x_1 \ldots x_{l'}$, $l' < l$ of a light subpath is also a valid light subpath, and store each vertex of the subpath in the data structure. These vertices are informally termed photons. Alternative formulations of this approach reason about photon mapping in terms of measuring the density of photons at the vertex $y_k$, and techniques based on photon mapping are sometimes called density estimators for this reason; similarly, $K$ can be referred to as the density estimation kernel.

**Bias.** The second issue encountered by this procedure is that we only want to connect subpaths whose end points are in close proximity. However, conditioning a connection this way must be accounted for in the PDF of the combined path: We must know the probability that any light subpath falls within the area of support of $K(x_l, y_k)$, which corresponds to integrating the light subpath PDF over the kernel. Computing this integral is generally not feasible; it can be estimated in certain cases [115], but the estimation is expensive and does not work in all path configurations. Instead, photon mapping makes the assumption that the PDF of the light subpath is constant over the support of the kernel, and the integration simplifies to a multiplication by the area of kernel support. This introduces bias: The PDF of the light subpath is essentially “blurred” over the area kernel, and this bias manifests as a blurring of lighting features in the image.

**Caustics.** A small, but significant optimization is to match the blurring of the PDF by simultaneously blurring the path contribution: Instead of evaluating the path contribution $f(z)$ over the combined path $z = x_0 \ldots x_{l-1} y_k \ldots y_0$, photon map-
ping instead approximates it with the contributions along the original subpaths, \( f(\mathbf{z}) \approx f(\mathbf{x}) \rho(\mathbf{y}_k) f(\mathbf{y}) \). Regardless of the location of \( \mathbf{y}_k \), we evaluate the original segment \( x_{l-1}x_l \), which avoids the potential source of variance discussed earlier. This variance could be significant: If the scattering function at \( x_{l-1} \) is very directional or even singular (e.g. a specular reflector), the contribution of the combined path \( f(\mathbf{z}) \) could exhibit arbitrarily high (or even infinite) variance. The approximate path throughput used in photon mapping always evaluates the segments that were actually sampled, leading to an enormous reduction in variance and can capture effects that no unbiased method can capture (e.g. reflected caustics).

Photon mapping with a fixed kernel size is biased and inconsistent. Recent progressive formulations \([57, 81]\) introduce a slightly altered version of photon mapping with progressively shrinking kernel sizes that remain consistent as light subpaths are added during the rendering progress.

Of special interest to this dissertation is photon mapping in participating media. Unlike surface transport, where the location of the next vertex on a path is uniquely determined by the location and direction (i.e. the ray) at the previous vertex, medium transport unlocks an additional degree of freedom where a vertex could reside at any point along the same ray. This can be exploited with special-tailored sensor queries \([72]\) and photons \([69]\) that account for all possible such locations, leading to beam estimators and photon beams that not only reduce variance, but bias as well. We will review these estimators in more detail and extend upon them in Chapter 5.

### 3.2.3 Markov Chain Monte Carlo

In our discussion of Monte Carlo methods in Section 3.1, we assumed the realizations \( Y_1, \ldots, Y_N \) of a Monte Carlo estimator to be independent. Markov Chain Monte Carlo (MCMC) is an interesting deviation from this norm, in which the samples of the estimator are dependent in a certain way; they form a *Markov chain*. 
We can view a Markov chain as a method for generating a sequence of random states \(X_1, X_2, \ldots\) in some state space \(\Theta\). At each state \(X_i\) in the chain, the next state \(X_{i+1}\) can be generated by drawing from a *transition kernel* \(q(X_{i+1} \mid X_i)\). This process is sometimes called memoryless, as each future state only depends on the present state, and not any state in the past.

Under relatively mild conditions\(^3\) on \(q\), the states of the Markov chain converge to a *stationary distribution* uniquely determined by \(q\), and regardless of the initial state \(X_1\).

The core idea of MCMC is to construct a Markov Chain whose stationary distribution is exactly proportional to the integrand of interest. After sufficiently many steps, the state of the Markov chain \(X_i\) will be distributed precisely according to the integrand, importance sampling it perfectly. The Metropolis-Hastings\(^4\) algorithm \([60, 100]\) achieves this by turning any transition kernel that does not give rise to the desired target distribution into one that does. Given a proposal kernel \(T(X_{i+1} \mid X_i)\) and a desired target distribution \(C\), the Metropolis algorithm uses the following transition kernel:

1. Sample a proposal state \(Y\) from the proposal distribution \(T(Y \mid X_i)\)
2. Compute acceptance probability \(r(Y \mid X_i) = \min \left\{ 1, \frac{C(Y)T(X_i \mid Y)}{C(X_i)T(Y \mid X_i)} \right\} \)
3. \(X_{i+1} = \begin{cases} Y & \text{with probability } r \\ X_i & \text{otherwise} \end{cases} \)

The crucial component of the Metropolis algorithm is the acceptance probability \(r\), which reshapes the proposal kernel into a transition kernel that leads to the desired stationary distribution. The probability is based on the idea of *detailed balance*: If \(X_i\)

---

\(^3\) In particular, the resulting Markov chain must be *ergodic*. A full discussion of ergodicity is out of scope for our purposes, but all the MCMC methods we will talk about in this dissertation satisfy it trivially by ensuring \(q\) is non-zero on the entire state space, i.e. any state can be reached in 1 step.

\(^4\) Although the Metropolis algorithm is named after Nicholas Metropolis, it is more accurately attributed to Arianna and Marshall Rosenbluth \([118]\).
and \( Y \) are both already distributed according to \( C \), then the rate of transitions from \( X_i \) to \( Y \) must be matched by the rate of transitions from \( Y \) to \( X_i \) in order to maintain the desired distribution.

The choice of proposal kernel is crucial to efficiency. If \( T \) is far from the ideal transition kernel, the Metropolis algorithm must “work harder” to reshape it into the desirable kernel, leading to low acceptance probability; the Markov Chain “stagnates” in the same state for many steps.

### 3.2.3.1 Metropolis Light Transport

Perhaps unintuitively, applying MCMC to the light transport problem is not straightforward. If we use the measurement contribution \( f_j \) as the target distribution and paths as states of the Markov chain, the Metropolis algorithm will indeed produce a sequence of paths \( \bar{x}_1, \bar{x}_2, \ldots \) that over time converge to the measurement contribution. We could thus write the standard primary Monte Carlo estimator

\[
Y = \frac{f_j(\bar{x}_i)}{p(\bar{x}_i)}
\]

and use the states of the Markov chain directly as Monte Carlo samples. What would the PDF \( p(\bar{x}_i) \) be, then? Assuming the Markov chain has reached stationary state, it is the normalized measurement contribution:

\[
p(\bar{x}_i) = \frac{f_j(\bar{x}_i)}{c_j} \quad \text{with} \quad c_j = \int p f_j(\bar{x}) \, d\mu(\bar{x}).
\]

Inserting into the primary estimator, we obtain

\[
Y = \frac{f_j(\bar{x}_i)}{f_j(\bar{x}_i) c_j^{-1}} = c_j;
\]
in other words, this MCMC algorithm is a perfect importance sampler with zero variance. However, evaluating it requires computing the normalization constant $c_j$, which is the integral we set out to solve in the first place. This estimator cannot work.

Metropolis Light Transport [135] (MLT) overcomes this challenge by observing that rendering involves solving many related integrals simultaneously. Almost always, we can decompose the sensor sensitivity of the $j$th sensor element (i.e. pixel) into a product of the total camera sensitivity $W_c$ that is the same for all pixels, and the reconstruction filter $h_j$ that selects the radiance seen by a particular pixel. The measurement equation can then be written as

$$I_j = \int_{\mathcal{P}} h_j(\mathbf{x}) f(\mathbf{x}) \, d\mu(\mathbf{x}), \quad (3.18)$$

where only the reconstruction filter $h_j$ changes between pixels $j$. MLT instantiates a Markov Chain with target distribution $f$, which leads to the primary estimator

$$Y = h_j(\mathbf{x}) \cdot c \quad (3.19)$$

for $I_j$. All pixels share the same Markov chain, and at each step of the chain, Eq. (3.19) is evaluated; this amounts to depositing $c$ at the pixel location where the current path resides. Rendering still requires computing $c$, but this factor is shared for all pixels in the image. This makes it feasible to estimate $c$ using a standard Monte Carlo estimator (e.g. bidirectional path tracing) and amortize its cost over all pixels.

In theory, the Markov chain will converge to the correct stationary distribution regardless of its initial state $\mathbf{x}_1$. However, poor initialization will affect the number of steps it takes until the chain reaches stationary state, and states early in the chain can be quite far from the desired distribution, which would lead to biased rendering. This is sometimes called start-up bias or the burn-in period. Veach et al. [135] proposes
an unbiased initialization step that avoids start-up bias entirely, and we refer to their work for a full discussion.

The critical component of the MLT algorithm is its choice of proposal distribution. Intuitively speaking, proposals must balance two conflicting goals: On the one hand, they must be sufficiently different from the current state to quickly explore the state space and avoid the chain “getting stuck” in local minima; on the other hand, they must be similar enough to the current state to allow local exploration of peaks in the target distribution $f$ and achieve high acceptance rates of proposals. MLT satisfies both of these goals by randomly selecting at each step between mutation strategies that make large changes to the current state, and perturbation strategies that make only small changes. The details of these strategies are not important for the rest of this dissertation, and we refer to Veach [131] for a full discussion.

### 3.2.3.2 Primary Sample Space MLT

Because Metropolis Light Transport operates directly on paths, it affords significant flexibility when it comes to designing mutations and perturbations. While this allows for very sophisticated and specialized perturbation strategies [59, 64, 79], the design of these perturbations, as well as computing their acceptance probabilities, is made tremendously complex by the structure of path space. Simultaneously, these perturbations only target specific physical effects (such as e.g. caustics), but modern scenes feature many interaction types that can even occur simultaneously on the same path.

Kelemen et al. [80] propose a reformulation of Metropolis Light Transport that allows significant simplification of the algorithm. It is based on the intuition that a classical rendering algorithm such as path tracing can be viewed as an abstract sampling scheme $S(\bar{u})$ that consumes a vector $\bar{u}$ of random numbers drawn uniformly, and returns a light path. Analogous to path space, the domain of these random vectors forms primary sample space $\mathcal{U}$. This space can be further decomposed over path
Consider now the following change of variables in the measurement equation:

\[ I = \int_X f(x) \, d\mu(x) = \int_{\mathcal{U}} f(S(\bar{u})) \left| \frac{dS(\bar{u})}{d\bar{u}} \right|^2 \, d\bar{u} = \int_{\mathcal{U}} \frac{f(S(\bar{u}))}{p(S(\bar{u}))} \, d\bar{u}. \tag{3.20} \]

The first step is simply a standard change of variables, where we have transformed the integral over paths into an integral over random numbers, using the sampling scheme \( S \) to map from one to the other. This incurs a Jacobian determinant to account for the change in measures. In the second step, we have used the fact that the Jacobian determinant of a sampling scheme is in fact the same as its inverse PDF - that is its definition. We have assumed here that primary sample space and path space have the same dimensionality. In general, this is not the case—sampling algorithms tend to consume many more random numbers than there are degrees of freedom on a path—and we will develop a more complete theory of transitioning between the two spaces in Chapter 6.

For notational convenience, from now on we will use the convention that any function previously defined over paths can be written with a primary sample space argument, using the relation \( f(\bar{u}) = f(S(\bar{u})) \).

Applying the Metropolis algorithm to the rightmost integral in Eq. (3.20) results in a remarkably simple algorithm. The state of the Markov chain is a vector of numbers between 0 and 1, and the state space is simply a product of unit intervals. The target function \( C(\bar{u}) = f(\bar{u})/p(\bar{u}) \) is simply the sample score, i.e. the integrand divided by the PDF of the sample, which is a quantity already tracked in practice by almost any sampling scheme \( S \). The symmetry and simplicity of the state space allow for a very straightforward proposal distribution: Kelemen et al. [80] propose to randomly choose between a mutation that replaces the current state \( \bar{u} \) with uniformly random
numbers (the “large step”), and a perturbation that adds a small amount of Gaussian noise to each element of \( \mathbf{u} \) (the “small step”).

The resulting rendering algorithm is Primary Sample Space MLT (PSSMLT). Given an existing rendering algorithm \( S(\mathbf{u}) \), PSSMLT can be implemented on top with relatively little effort. In addition, PSSMLT is able to take advantage of all sophisticated importance sampling techniques employed inside \( S \). As a result, no specialized perturbation strategies tailored to specific lighting effects are required, and the proposal distribution in PSSMLT is remarkably simple. This “black box” view of the path generation process in \( S \) is not without cost, however, and as we will see in the following section, allowing slightly finer-grained control over paths can improve variance significantly.

### 3.2.3.3 Multiplexed MLT

One advantage of PSSMLT is that it can be combined with any existing rendering algorithm. The target function \( f(\mathbf{u})/p(\mathbf{u}) \) automatically “steers” the sampling algorithm towards regions that it does not sample well. In practice, PSSMLT is most often combined with bidirectional path tracing [80] owing to the already robust nature of BDPT; adding Metropolis can “iron out” the few remaining poorly sampled paths.

Combining PSSMLT and BDPT is not without issues, however. Internally, BDPT uses not one, but \( k + 2 \) different sampling schemes \( S_0(\mathbf{u}), \ldots, S_{k+1}(\mathbf{u}) \) to construct paths of length \( k \) from a concatenation of shorter sensor/light subpaths of different lengths. When evaluating \( S(\mathbf{u}) \), BDPT internally constructs many (up to the square of the maximum path length) paths simultaneously and takes their weighted combination. In practice, most of the sampling techniques used by BDPT perform relatively poorly, and the majority of these paths receive low to zero weights from multiple importance sampling, but still add to the computational expense.
Multiplexed MLT [56] (MMLT) addresses this issue by exposing all internal sampling schemes to the Metropolis sampler. MMLT simulates a separate Markov chain for each path length \( k \) in order to disambiguate the path length, and uses the first coordinate of the state \( \bar{u} \) to select which of the \( k + 2 \) available sampling schemes to use for mapping states to paths:

\[
    j = \lfloor u_0 \cdot (k + 2) \rfloor. \tag{3.21}
\]

MMLT then uses the modified target distribution

\[
    C_j(\bar{u}) = m_j(\bar{u}) f_j(\bar{u}) / p_j(\bar{u}), \quad \text{where} \quad m_j(\bar{u}) = \frac{p_j(\bar{u})}{\sum_{i=0}^{k+1} p_i(\bar{u})} \tag{3.22}
\]

is the MIS weight assigned by BDPT to the sampling scheme \( S_j(\bar{u}) \) for the current path. We use the shorthand \( f_j(\bar{u}) = f(S_j(\bar{u})) \) to refer to path space quantities seen through a particular mapping \( S_j \).

With these changes, the Metropolis sampler can not only exert control over which regions of primary sample space to explore, but also which mapping to use to transition from primary sample space to path space. The modified target distribution automatically steers Metropolis towards techniques \( j \) that are assigned a large weight by MIS, concentrating computational effort only on strategies that contribute meaningfully to the image.

MMLT imposes a new structure on primary sample space, where the space is divided into \( k + 2 \) strata along the first dimension. Each of these strata corresponds to path space seen through a different mapping \( S_j \). Unfortunately, switching from one mapping to the other is highly discontinuous: The different sampling schemes in BDPT are not bound to be related, and even if all other dimensions of \( \bar{u} \) are fixed, changing \( j \) will generally lead to a completely different path in path space. This means that there are no “perturbations” when a change of sampling techniques is
involved, as the path and subsequently the path contribution change dramatically, leading to low acceptance rates of such changes and poor local explorations. We address these issues in Chapter 6 with a new type of perturbation.
Part II

METHODS

It’s not worth doing something unless you were doing something that someone, somewhere, would much rather you weren’t doing.

— Terry Pratchett
A RADIATIVE TRANSFER FRAMEWORK FOR
NON-EXPOENTIAL MEDIA

A central assumption made in the equations of radiative transfer we have reviewed so far, and indeed core to our medium representation in Section 2.2.2, is that the scatterers in a participating medium are statistically independent from each other. This gives rise to the classical exponential transmittance.

While this assumption has proved useful in many applications, this memoryless model of collisions with scattering particles cannot represent media which exhibit any form of correlation between the location of scatterers. In this chapter, we will introduce a new model of radiative transfer that can capture correlated media with non-exponential transmittance, and allows for a richer space of volumetric appearance.

We will begin this chapter by first making a case for why non-exponential transmittance is a useful tool (Section 4.1), and show in a simple experiment how non-exponentiality arises naturally through correlated scatterers. One may be tempted to model non-exponentiality by simply substituting transmittance in the measurement equation with a non-exponential term, but we show why this straightforward approach cannot work (Section 4.2). We then proceed to derive a non-exponential path integral as a statistical average of microscopic statistical heterogeneity in Section 4.3. We would go amiss without acknowledging the rich history of non-exponential transport in both graphics and other fields, and discuss how our proposed framework relates to other works in Section 4.3.3. To leverage the flexibility of our new path integral, we use the mathematical formalisms of stochastic Gaussian processes and
The case for non-exponential transport

All participating media in reality exhibit correlation between scatterers to some extent. The simplest mode of correlation is that real scatterers must have some physical size, and no two scatterers can occupy the same volume of space. This imposes a minimum distance of separation between scatterers, and already violates the independence assumption.

An important attribute of correlation is one of scale. Non-exponentiality becomes significant if correlations rise to the scale of the mean-free path of the medium [32], where positive correlations [17, 84] lead to slower-than-exponential, and negative correlations [122] to faster-than-exponential free-flights respectively.

In the earlier example, the effect of volume-exclusion on the particle distribution is only observable within a few particle radii of a scatterer; at larger scales, the same particles would still appear independently random. Much longer-range correlations are caused by inter-particle forces, which have been widely studied in the class of colloids, i.e. mixtures of insoluble microscopic particles dispersed in another substance. This covers a wide variety of materials such as foods (e.g. milk, honey, soups, juices), agents (gels, foams, oils, ink, paint, toothpaste), gases (mist, dust, fog, steam, smoke, clouds), outer space (stellar clouds, planetary rings, comets) and even some glasses and plastics. Interactions between these particles is dictated by a variety of forces, such as electrostatic- or van der Waals forces. The inter-particle correlations caused by these forces has been widely reported in literature [7, 48, 52, 65, 117, 123, 146]. For the specific case of clouds, growing evidence from atmospheric sciences shows
that correlations in the positions of water droplets exist at scales ranging at least from centimeters to kilometers \([33, 83]\). While long-range correlations could be represented with e.g. a voxelized, heterogeneous classical medium, within each voxel the medium is homogenized into a classical medium exhibiting no correlations. Ensuring no correlations are homogenized away and lost in this process would require prohibitively small voxels in most cases (e.g. clouds). A non-exponential transport model would allow for much coarser homogenization by folding correlations smaller than the voxel size into the free-flight distribution.

Another source of non-exponentiality of increasing importance in graphics arises when media are used as a tool for level-of-detail. Highly detailed scene geometry located far from the camera causes a computational burden and aliasing artifacts due to a mismatch with the level of detail actually observable in the rendering. It is therefore highly desirable to prefilter the scene representation to discard higher frequencies while preserving the overall appearance. Approaches to this problem are diverse \([20]\), but recent work has shifted increasingly toward using media to represent surfaces \([95, 96, 136]\). However, surfaces inherently exhibit strong correlation and cannot be represented accurately with exponential media. Highly related is the problem of granular media rendering \([99, 101, 104]\), in which densely packed arrangements of small scattering grains (e.g. sand, sugar, snow) are to be rendered. Prevalent solutions in this space replace the explicit grains with a classical medium with numerically fitted parameters to simplify transport; however, this solution is exclusively used for strongly indirect transport because it is only an approximate match. Part of the mismatch with the ground truth is because of the dense packing of scatterers, due to which free-flights between scattering events show significant correlation not faithfully represented by an exponential model. Finally, there is the problem of level-of-detail for media themselves: Replacing a high-resolution voxelized medium with a lower-resolution one without affecting overall appearance is a challenging problem. Simply spatially averaging the media properties to obtain the
4.1 THE CASE FOR NON-EXPOENTIAL TRANSPORT

Figure 4.1: Top row: We show media with discrete scatterers (top half) and continuous densities (bottom half) of different distributions (a)–(d), and the average transmittance measured in these media as a function of mean free paths (e). Independently placed scatterers ((a), white noise) lead to the classical exponential transmittance. Negatively correlated scatterers ((b), blue noise) lead to faster-than-exponential extinction. Positively correlated scatterers ((c)–(d), pink/red noise) lead to slower-than-exponential extinction.

A lower-resolution medium will lead to undesirable results, as the original medium is non-exponential (due to heterogeneity) when measured at the scale of a voxel in the lower-resolution medium.

To illustrate these effects, we show how diverse non-exponential transmittances arise naturally from correlated scatterers. In Fig. 4.1 we perform a simple Monte Carlo experiment where we trace photons with random origins and directions through 2D media consisting of explicitly modeled absorbing particles that are: (a) statistically independent, (b) negatively correlated, and (c,d) positively correlated. We gather statistics about the averaged transmittance (e) by tabulating along the horizontal
axis the fraction of photons that survive for a given distance. In each case the average number of particles is identical, and it is the statistical correlations that give rise to different light attenuation behavior. Intuitively, particles in positively correlated media (c,d) “clump” together and leave larger gaps than expected, so photons that traverse these gaps skew the free-flight distribution towards the tail, allowing light to penetrate further on average. The opposite happens in negatively correlated media. Ignoring these correlations and instead assuming statistical independence leads to an inaccurate estimation of light transport with notably different visual appearance.

4.2 Non-exponentiality is Non-trivial

It is not immediately obvious why non-exponential transport requires a new transport framework. It appears straightforward to substitute a non-exponential function for the transmittance $T_r(\tau) = e^{-\tau}$ in the measurement equation (Eq. (2.34)). In this section, we show a simple proof for why this approach must violate energy conservation.

In the following, we consider a pencil beam of light starting at location $x$ traveling through a participating medium in direction $\omega$ that intersects a surface at point $x_z$.

The radiance leaving the beam origin is $L_o(x, \omega)$, and the radiance received by points on the beam decreases with distance as a result of extinction by the medium. The radiance must then be distributed between two terms: the amount of light that collides with scatterers in the medium, and the remaining fraction that reaches the
surface. The sum of both terms must equal $L_0(x, \omega)$—otherwise, energy is either lost or gained along the way. This results in the following constraint:

$$L_0(x, \omega) = L_0(x, \omega) \cdot \text{Tr}(\tau(x, x_z)) + \int_0^Z L_0(x, \omega) \cdot \text{Tr}(\tau(x, x_s)) \sigma(x_s) \, ds,$$

(4.1)

The above equation can be simplified by dividing out $L_0(x, \omega)$:

$$1 = \text{Tr}(\tau(x, x_z)) + \int_0^Z \text{Tr}(\tau(x, x_s)) \sigma(x_s) \, ds.$$

(4.2)

Using the fact that $d\tau/ds = \sigma(x_s)$, we can perform a change of variable from $ds$ to $d\tau$ to obtain

$$1 = \text{Tr}(\tau(x, x_z)) + \int_0^{\tau(x, x_z)} \text{Tr}(\tau) \, d\tau.$$

(4.3)

Any transmittance function $\text{Tr}$ that does not satisfy the above constraint violates energy conservation.

Eq. (4.3) is an ordinary differential equation of the form $1 = f'(x) + f(x) - f(0)$. The only solutions that satisfy it are expressed by $\text{Tr}(\tau) = c \cdot e^{-\tau}$. In other words, only an exponential transmittance can satisfy energy conservation in the classical path integral. The naive solution of substituting a non-exponential function for the transmittance in the classical path integral will invariably lead to energy loss or, worse, energy gain, and a more principled approach is needed to support non-exponentiality while maintaining energy conservation.

## 4.3 A Non-Exponential Path Integral

Similar to the classical path integral, we formulate our non-exponential rendering problem as the average behavior of light interacting with a collection of (correlated)
stochastic scatterers. Each realization $\mu$ encompasses a concrete configuration of scatterers, and we take expectations over all possible such realizations. Before we proceed, we must first pick a model of how each realization of these scatterers is represented: We can either choose to model the scatterers as a discrete point distribution (Fig. 4.1, top), or describe each realization as a continuous, heterogeneous classical medium (Fig. 4.1, bottom). Both models result in the same theory. We begin with the continuous model, as it is more mathematically convenient, but give a sketch for a derivation from the discrete model in Appendix A.1.

We use the classical measurement equation to describe transport in each realization, and will use the subscript $\mu$ to refer to quantities specific to each realization. For example, $\sigma_\mu(x)$ refers to the extinction coefficient of realization $\mu$. Quantities associated with surfaces are assumed to be invariant with respect to the realization. In this model, the measured radiance in each realization is

$$I_\mu = \int_{\mathcal{P}} L_e(x_0)g_\mu(\vec{x})W_e(x_k)\,d\mu(\vec{x})$$

with the realization-specific path throughput

$$g_\mu(\vec{x}) = \Sigma_\mu(x_0) \left[ \prod_{i=1}^{k-1} \rho_\mu(x_i)\Sigma_\mu(x_i) \right] \left[ \prod_{i=0}^{k-1} \text{Tr}_\mu(x_i, x_{i+1})G(x_i, x_{i+1}) \right] \Sigma_\mu(x_k).$$

Our goal is to compute the expected value, or ensemble average $\langle I_\mu \rangle$ of the measurement over all realizations. In the following, we assume that emission and sensing of the scatterers in the medium is invariant across realizations, i.e. $L_e(x_0)$ does not depend on the realization directly. However, emission and sensing are still correlated with the other scatterers: The density of scatterers (measured via $\Sigma_\mu(x_0)$) still affects the amount of emission at $x_0$, and similar for sensing.
This reduces the problem to computing the average path throughput:

\[
\langle g_\mu(x) \rangle = \langle \Sigma_\mu(x_0) \left[ \prod_{i=1}^{k-1} \rho_\mu(x_i) \Sigma_\mu(x_i) \right] \left[ \prod_{i=0}^{k-1} \text{Tr}_\mu(x_i, x_{i+1}) G(x_i, x_{i+1}) \right] \Sigma_\mu(x_k) \rangle .
\] (4.6)

This ensemble average captures all correlation effects in the medium, but is impractical to compute in general: All terms of the path throughput above depend on the realization \( \mu \) and are correlated with each other, which makes simplification near impossible without making further assumptions.

Simplifying assumptions are common in prior works on deriving non-exponential transport equations. Of particular interest to us is the work of Larsen et al. [91], who derive a generalized linear Boltzmann equation (GBE) that can describe non-exponential transport. Although their model has several limitations that make it impractical for use in graphics, we take inspiration from their approach and apply some of their assumptions to our more general problem.

We defer a full comparison between our model and that of Larsen et al. to Section 4.3.3, after our model is fully derived. In the interim, we proceed with our derivation in three steps: We first import two critical assumptions from the Larsen et al. model and apply them to Eq. (4.6) to obtain a new non-exponential path integral. This path integral is not reciprocal in the traditional sense, and we address this issue in Section 4.3.1 to derive a fully reciprocal version. Up until this point, we support statistical heterogeneity in its most general form, which makes practical rendering difficult; to address this, we introduce a practical heterogeneity model in Section 4.3.2 that preserves non-exponentiality while enabling efficient rendering algorithms for the heterogeneous case.

Before we can proceed with the derivation of our proposed path integral, we first need to formally define the notion of an ensemble average. We associate with each realization \( \mu \) of the medium a probability measure \( dP(\mu) \) that describes its likelihood.
An ensemble average of an arbitrary function $f_\mu(x)$ is then a measurement over $\mathcal{R}$, the space of all realizations (the ensemble):

$$\langle f_\mu(x) \rangle = \int_\mathcal{R} f_\mu(x) \, dP(\mu). \quad (4.7)$$

The ensemble average preserves all properties of the integral, such as linearity.

Returning to the problem of computing $\langle g_\mu(\vec{x}) \rangle$, we now introduce our first assumption: The scattering function is uncorrelated with the realization $\mu$. The linearity property of $\langle \cdot \rangle$ then allows us to move these terms outside of the ensemble average:

$$\langle g_\mu(\vec{x}) \rangle \approx k^{-1} \prod_{i=1}^{k-1} \rho(x_i) \prod_{i=0}^{k-1} G(x_i, x_{i+1}) \left( \Sigma_\mu(x_0) \prod_{i=0}^{k-1} \text{Tr}_\mu(x_i, x_{i+1}) \Sigma_\mu(x_{i+1}) \right). \quad (4.8)$$

This simplifies the problem to computing the average transmittance and extinction coefficient across the entire path. For notational convenience, we have also moved the geometry term, which is independent of the realization.

The second assumption we import in a modified form from Larsen et al. is that photon free-flights only depend on the distance to the previous event. In other words, the “memory” of each photon does not extend beyond a single segment of the path; intuitively, the photon “jumps” into a different realization every time it interacts with the medium. This assumption effectively decorrelates segments of the path from each other, and this allows us to compute the ensemble average independently for each segment instead of over the path as a whole (Fig. 4.2, bottom):

$$\left\langle \Sigma_\mu(x_0) \prod_{i=0}^{k-1} \text{Tr}_\mu(x_i, x_{i+1}) \Sigma_\mu(x_{i+1}) \right\rangle \approx \langle \Sigma_\mu(x_0) T_\mu(x_0, x_1) \rangle \prod_{i=1}^{k-1} \langle T_\mu(x_i, x_{i+1}) \rangle. \quad (4.9)$$

Here we have swapped the order of the ensemble average and the product, and substituted a transport kernel $T_\mu$ for the segment term. What should this term be? If
free-flights depended purely on the distance to the previous event, then this function would simply be $\text{Tr}_\mu \times \Sigma_\mu$, the classical segment term.

Unfortunately however, the classical segment term is not sufficient here; a transport kernel defined this way would lead to a non-reciprocal transport framework: First note that the classical segment term actually represents two different functions, depending on the end point of the segment. If $x_{i+1}$ lies in the medium, the segment term becomes the free-flight PDF $p_\mu$; if it does not, $\Sigma_\mu$ becomes 1 and the segment term reduces to the transmittance $\text{Tr}_\mu$ instead. Now consider the case of transport between a surface point $x$ and a point in the medium $y$: The surface-to-medium transport $T_\mu(x, y)$ would reduce to the free-flight PDF $p_\mu$, whereas the medium-to-surface transport would result in the transmittance $\text{Tr}_\mu$ instead, meaning that energy flow is asymmetric when the direction is reversed.

To solve this problem, we modify the assumption of Larsen et al. to include an additional piece of “memory”, which is whether the previous interaction was with a scattering particle, or was in “free space” (i.e. a location not constrained to coincide with a scatterer, such as a surface).
This results in four different functions (see Fig. 4.3) that represent the four possible combinations of $x_i$ and $x_{i+1}$ coinciding with a particle or lying in free-space. These four functions form the ensemble transport kernel:

$$T(x, y) = \langle T_{\mu}(x, y) \rangle = \begin{cases} 
FF(x, y) & \text{if } x \in A \land y \in A \\
fp(x, y) & \text{if } x \in A \land y \in V \\
PF(x, y) & \text{if } x \in V \land y \in A \\
pp(x, y) & \text{if } x \in V \land y \in V,
\end{cases}$$

(4.10)

where we have labeled the functions according to the type of transport they represent, i.e. “free space-to-free space”, “free space-to-particle”, “particle-to-free space” and “particle-to-particle”. It is worth noting that $fp$ and $pp$ represent conceptually different quantities (with different units) than $FF$ and $PF$, which makes the units of the transport kernel $T$ change depending on the location of the segment end points. We will return to this topic after deriving concrete expressions for the four functions, and resolve this inconsistency when we ensure reciprocity of our transport equations.

To obtain concrete expressions for these four functions, we return to the right-hand side of Eq. (4.9). We will still use the classical segment term for $Tr_{\mu}(x_i, x_{i+1})$, but will use a modified ensemble average. If $x_i$ coincides with a scatterer, then the regular ensemble average is no longer appropriate: The ensemble may include many realizations of the medium in which there is no scatterer at $x_i$. Instead, we need to compute an average conditioned on $x_i$ coinciding with a scatterer.

The probability density of encountering a scatterer at $x_i$ is the number density $\eta_{\mu}(x_i)$. To express the conditioned ensemble average, we need to reshape the probability measure to account for the density of particles in one realization compared to the rest of the ensemble, i.e. $\eta_{\mu}(x_i) / \eta(x_i) \cdot dP[\mu]$. Assuming that the cross-section of the particles does not depend on the realization, we can instead express the change
of measures via the more convenient extinction coefficient to obtain the conditioned ensemble average:

\[ \langle f(x) \rho_\mu(x) \rangle \quad \text{with} \quad \rho_\mu(x) = \frac{\sigma_\mu(x)}{\sigma(x)}. \quad (4.11) \]

With this tool, we can now define the four transport functions:

\[ \text{FF}(x, y) = \langle \text{Tr}_\mu(x, y) \rangle \quad (4.12a) \]
\[ \text{fp}(x, y) = \langle \text{Tr}_\mu(x, y) \sigma_\mu(y) \rangle \quad (4.12b) \]
\[ \text{PF}(x, y) = \langle \text{Tr}_\mu(x, y) \rho_\mu(x) \rangle \quad (4.12c) \]
\[ \text{pp}(x, y) = \langle \text{Tr}_\mu(x, y) \sigma_\mu(y) \rho_\mu(x) \rangle. \quad (4.12d) \]
We can now also simplify the first term of the right-hand-side of Eq. (4.9):

\[
\langle \Sigma_\mu(x_0) T_\mu(x_0, x_1) \rangle = \Sigma(x_0) \left( \frac{\Sigma_\mu(x_0)}{\Sigma(x_0)} T_\mu(x_0, x_1) \right) \\
= \Sigma(x_0) \cdot \begin{cases} 
\langle \text{Tr}_\mu(x_0, x_1) \Sigma(x_1) \rho_\mu(x) \rangle & \text{if } x_0 \in V \\
\langle \text{Tr}_\mu(x_0, x_1) \Sigma(x_1) \rangle & \text{if } x_0 \in A
\end{cases} \\
= \Sigma(x_0) T(x_0, x_1)
\] (4.13)

With this, our path integral becomes

\[
\langle g_\mu(x) \rangle \approx \Sigma(x_0) \prod_{i=1}^{k-1} \rho(x_i) \prod_{i=0}^{k-1} G(x_i, x_{i+1}) \prod_{i=0}^{k-1} T(x_i, x_{i+1}).
\] (4.16)

It is not immediately clear that this throughput is reciprocal: It appears that the transport kernel \( T \) gives different results if the segment is reversed and thus leads to non-reciprocity. However, this is only true at first glance and can be resolved with a simple regrouping of terms, which we show in Section 4.3.1. However, before then, it is worth analyzing the four different functions of \( T \) in more detail and develop some intuition for their meaning.

As noted earlier, \( f_p \) and \( p_p \) represent conceptually different quantities than \( F_F \) and \( P_F \). Here, \( pp(x_i, x_{i+1}) \) and \( fp(x_i, x_{i+1}) \) are averages of free-flight PDFs and are themselves free-flight PDFs (with unit \( m^{-1} \)): They correspond exactly to the density of particle-to-particle and free-space-to-particle flights. On the other hand, \( PF(x_i, x_{i+1}) \) and \( FF(x_i, x_{i+1}) \) are averages of transmittances and are themselves (unitless) transmittances.

In the classical path integral, the transmittance and free-flight PDF are tightly related (Eq. (2.8)). Because the functions we introduced are themselves averages of the
classical transmittance and free-flight PDF, and the averaging operator is linear, we can easily prove that the same holds for the averaged functions:

\[ \text{FF}(x, x_t) = \langle \text{Tr}_\mu(x, x_t) \rangle \]  
\[ = \left\langle \int_{x_t}^{\infty} p_\mu(x, x_s) \, ds \right\rangle \quad \text{from Eq. (2.8)} \]  
\[ = \int_{x_t}^{\infty} \langle p_\mu(x, x_s) \rangle \, ds \quad \text{from linearity of } \langle \cdot \rangle \]  
\[ = \int_{x_t}^{\infty} f_p(x, x_s) \, ds \quad \text{from Eq. (4.12b),} \]

and analogously for PF and pp. Perhaps more surprisingly, we can also prove that PF and fp are directly related. Intuitively, \( f_p(x, y) \) represents the PDF of colliding with a scatterer at \( y \) starting from free-space point \( x \), whereas \( \text{PF}(x, y) \) represents the probability of reaching free-space point \( y \) starting from a scatterer at \( x \). These functions describe reverse events in a sense, and indeed we can show

\[ f_p(x, y) = \langle \text{Tr}_\mu(x, y) \sigma_\mu(y) \rangle \]  
\[ = \sigma(y) \cdot \left\langle \text{Tr}_\mu(y, x) \frac{\sigma_\mu(y)}{\sigma(y)} \right\rangle = \sigma(y) \cdot \langle \text{Tr}_\mu(y, x) \rho_\mu(y) \rangle \]  
\[ = \sigma(y) \cdot \text{PF}(y, x) \]

These four functions are not arbitrary; they represent probabilities (densities) of events between two points, and are tightly linked to each other—given one function, all three others can be derived. The relationships between them are a direct result of averaging transmittances and free-flight PDFs (4.20) and performing unconditioned averages or averages correlated with scatterer locations (4.23). We illustrate these relationships and their inverses in Fig. 4.3.
\section*{4.3.1 Reciprocity}

So far, our transport kernel appears not reciprocal, i.e. $T(x, y) \neq T(y, x)$ in the general case. There are two reasons for this: Free-flight PDFs (such as pp) are not generally symmetrical when their arguments are exchanged, and swapping the end-points of a surface-to-medium segment causes a different case statement to be evaluated (fp in one case, PF in the other).

However, we will show that this apparent problem can be resolved through a simple rearranging of terms. We proceed by deriving a reciprocal transport kernel through mathematical manipulation and then offer an intuitive interpretation of this new kernel.

Consider the modified transport kernel $\tilde{T}$:

$$\tilde{T}(x, y) = \frac{T(x, y)}{\Sigma(y)} = \begin{cases} 
FF(x, y) & \text{if } x \in \mathcal{A} \land y \in \mathcal{A} \\
\frac{fp(x,y)}{\sigma(y)} = PF(y, x) & \text{if } x \in \mathcal{A} \land y \in \mathcal{V} \\
PF(x, y) & \text{if } x \in \mathcal{V} \land y \in \mathcal{A} \\
\frac{pp(x,y)}{\sigma(y)} & \text{if } x \in \mathcal{V} \land y \in \mathcal{V}.
\end{cases}$$ \hspace{1cm} (4.24)

Different to before, $\tilde{T}$ will evaluate PF for a segment that has one end-point on a surface and another in the medium, regardless of direction. Additionally, $\tilde{T}$ evaluates $pp(x, y)/\sigma(y)$ for a segment with both end points in the medium. We can easily prove that this term is equivalent to averaging the transmittance along a segment, conditioned on both end points lying on a particle:

$$pp(x, y)/\sigma(y) = \frac{\langle \rho_\mu(x)Tr_\mu(x,y)\sigma_\mu(y) \rangle}{\sigma(y)} = \frac{\langle \rho_\mu(x)Tr_\mu(x,y)\rho_\mu(y) \rangle}{\sigma(y)}$$ \hspace{1cm} (4.25)
We can interpret this as an ensemble averaged transmittance, conditioned on both \( x \) and \( y \) coinciding with a scatterer. This means that \( \tilde{T} \) evaluates only averages of transmittance for all four possible configurations of a segment. The classical transmittance is reciprocal, and so are averages of transmittances; hence, \( \tilde{T} \) is reciprocal. This also ensures the units of \( \tilde{T} \) are consistent, as only a (unitless) transmittance is evaluated in all cases.

To complete the derivation, we use the relation \( T(x, y) = \Sigma(y)\tilde{T}(x, y) \) in Eq. (4.16) to obtain our proposed reciprocal, non-exponential path throughput:

\[
\langle g_{\mu}(x) \rangle \approx \Sigma(x_0) \left[ \prod_{i=1}^{k-1} \rho(x_i)\Sigma(x_i) \right] \left[ \prod_{i=0}^{k-1} \tilde{T}(x_i, x_{i+1})G(x_i, x_{i+1}) \right] \Sigma(x_k). \tag{4.26}
\]

We have rearranged the \( \Sigma \) terms to obtain an equation that looks near identical to the classical path throughput. The only difference to the exponential case is the replacement of transmittance \( T \) with our modified transport kernel \( \tilde{T} \). This makes it straightforward to simulate our proposed non-exponential transport model with existing (bidirectional) rendering methods. We describe the necessary changes to existing renderers in detail in Section 4.5.

### 4.3.2 Heterogeneity

At the beginning of this section, we formulated the original averaging problem in a statistically heterogeneous medium. As a result of this, all equations so far explicitly support heterogeneity in its most general form—for example, the macroscopic density could vary, or the correlation between scattering particles could change from one part of the medium to the next.

Although this generality can be useful, it also means that the transport kernel \( \tilde{T}(x, y) \) is a 6D function. In the general case, this makes it impractical to derive analytical forms of \( \tilde{T} \) or to tabulate it numerically, and rendering with this form of the
The path integral is difficult. To address this, we introduce an explicit decomposition of the medium into a 1D transport kernel and a 3D density field that enables practical rendering methods while preserving non-exponentiality.

Our current model and the model of Larsen et al. can be viewed as lying on two opposite sides of a spectrum: The Larsen et al. model assumes a fully homogenized model with identical statistics everywhere, whereas our model makes no assumptions about how correlations are distributed spatially.

Assuming statistical homogeneity is convenient, because ensemble-averaged transmittances and free-flight PDFs become functions of only distance. Parametric models of transmittance can be readily derived for certain classes of stochastic media (Section 4.4) if they are statistically homogeneous, and for all other media, the required 1D functions can be easily tabulated numerically. On the other hand, heterogeneous media find wide-spread use in graphics, and any path integral not supporting such media is heavily limited.

This motivates us to seek a middle ground between the two models that combines the benefits of both: We decompose the description of the medium into a heterogeneous, macroscopic density $\sigma(x)$ and a 1D transport kernel $T(t)$.

In other words, we assume that the medium is statistically heterogeneous, but that the underlying correlations are the same throughout the medium. Our model imposes that for a medium with unit macroscopic density ($\sigma(x) = 1$), averages of transmittance and free-flight PDF become functions only of distance:

$$\text{FF}(x, x_t) = \text{FF}(t) = \langle \text{Tr}(t) \rangle$$

$$\text{fp}(x, x_t) = \text{fp}(t) = \langle \text{Tr}(t) \sigma(\mu(x_t)) \rangle$$

$$\text{PF}(x, x_t) = \text{PF}(t) = \langle \text{Tr}(t) \rho(\mu(x_t)) \rangle$$

$$\text{pp}(x, x_t) = \text{pp}(t) = \langle \text{Tr}(t) \sigma(\mu(x_t)) \rho(\mu(x_t)) \rangle$$
We can view these functions as forming a “base” transport kernel $\tilde{T}_{\text{base}}(t)$ that captures only correlations, but not heterogeneity.

For a medium with non-unit macroscopic density but the same correlations as the medium above, we express its transport kernel in terms of the base kernel and the macroscopic optical depth:

$$
\tilde{T}(x_i, x_{i+1}) = \tilde{T}_{\text{base}}(\tau_i) = \begin{cases} 
\text{FF}(\tau_i) & \text{if } x_i \in A \land x_{i+1} \in A \\
\text{pp}(\tau_i) & \text{if } x_i \in V \land x_{i+1} \in V \\
\text{PF}(\tau_i) & \text{else,}
\end{cases}
$$

(4.28)

with $\tau_i = \tau(x_i, x_{i+1})$. We obtained this equation by inserting the equivalent of (4.24) for $\tilde{T}_{\text{base}}$. Because $\text{pp}$ is a PDF, it incurs a Jacobian factor of $\sigma(x_{i+1})$ from the transformation. However, this factor cancels the division by $\sigma$ from (4.24), leaving us with a remarkably simple transport kernel.

This decomposition can be interpreted as separating the medium into explicitly modeled density variations and statistically modeled density fluctuations that exist beyond what is modeled explicitly. This model has several advantages: Reducing the transport kernel to 1D makes it practical to estimate it numerically or derive it analytically. Additionally, the decomposition of the medium into a heterogeneous density and a “plug-in” function describing correlations allows us to easily superimpose non-exponentiality onto existing heterogeneous media without additional effort. Finally, relating the microscopic and macroscopic model via the optical depth enables practical sampling methods that are even unbiased in some special cases (Section 4.5).

**Physical interpretation.** This heterogeneity model has a simple physical interpretation, which we illustrate in Fig. 4.4. The left column in this figure represents an
Figure 4.4: We express volumetric micro-roughness $\sigma_\mu$ (left) statistically and model macro-scale extinction $\sigma$ explicitly (black, homogeneous: center, heterogeneous: right) and combine them (blue). When we pass the base transport functions through macroscopic optical depth $\tau(s)$, the statistical fluctuations/correlations that $\sigma_\mu$ (left) represents are scaled vertically and squeezed horizontally (middle, right) by the local extinction coefficient $\sigma(s)$.

instance of the “base” medium, which is statistically homogeneous and has a mean extinction of 1. The middle column of this figure represents an instance of a medium with the same correlations, but a mean extinction of 2. Through Eq. (4.28), we can relate transmittance in this medium to transmittance in the base medium via the optical depth; this can be interpreted as “squishing” the base medium horizontally and “stretching” it vertically, so that it has twice the amplitude and twice the frequency. In the general case of a macroscopically heterogeneous medium (right column), the amount of this stretching is determined by the local macro-scale extinction coefficient $\sigma(x)$.

## 4.3.3 Related Non-Exponential Models

Non-exponential transport in graphics has only received attention in recent years, but fields such as neutron transport or atmospheric sciences have a rich history of correlated media research. Now that our path integral formulation is established, we are able to compare our work to the ideas of prior and concurrent work.
Atmospheric Sciences. The atmospheric science community has developed many statistical models [17, 32, 34, 35, 84, 122] to explain how observable correlations of water droplets in clouds [33, 83] lead to non-exponential aggregate transport behavior.

Neutron Transport. Within the field of neutron transport, Larsen and Vasquez (2011, 2014, 2014) recently derived generalized RTE-like models for neutron transport that allows non-exponential, angle-dependent free flight distributions to model correlations observed in pebble-bed reactors. Non-exponential behavior also arises when there is “cross-talk” between neutrons (photons) of different energy-levels (wavelengths). This is often ignored in graphics – unless fluorescence/in-elastic scattering [54, 67] needs to be considered – but is quite common in neutron transport where multi-energy simulations are standard practice. d’Eon [38] provides an excellent overview of previous work on non-exponential free-flights outside of graphics.

Discussion. Unfortunately, leveraging the methods developed in either of these fields in graphics is challenging. Methods from atmospheric sciences only consider bulk transport at large scales, and fully homogenize the medium into lower-dimensional “slabs” of media that only resolve radiometric quantities over one dimension (i.e. distance from ground). As for neutron transport, the quantity of interest—neutrons—has such penetration power that fully opaque surfaces do not exist, and methods from this field generally do not include treatment of surfaces. In addition, spatially resolved heterogeneity that go beyond compositions of homogeneous media are of high importance in graphics to model effects such as smoke or clouds, but are not generally of interest in neutron transport.

For these reasons, we cannot directly use any of these methods. However, there is much to be learned from these works, and we import some of the ideas of these prior works. For example, although their model of radiative transfer does not fit our
constraints, the fractal variability model proposed by Davis and colleagues (2011, 2014) nonetheless forms a useful parametric transmittance that can be plugged into our framework (Section 4.4).

Of particular interest is the General Boltzmann Equation (GBE) of Larsen et al. [91]. Concurrently to our work, this method has seen several recent extensions to support different forms of heterogeneity, surfaces, and reciprocity, and we discuss these works in more detail below. Our method, too, is based to some extent on the GBE, and we import some of the same assumptions to derive our path integral. In the case of an infinite, statistically homogeneous medium without surface boundaries, our proposed path integral is mathematically equivalent to their GBE; this is because then all path vertices lie in the medium and our transport kernel only evaluates $p_p$, the particle-to-particle free-flight PDF, which is the same function used in the GBE. Our main contributions lie in creating a practical framework for graphics: We introduce surfaces in a reciprocal manner, support full heterogeneity, and provide a path integral for easy integration into existing renderers.

**Concurrent Work.** Concurrent to our work, d’Eon [39] extend the work of Larsen et al. [91] to include surfaces while ensuring weak reciprocity of the resulting RTE. He achieves this by deriving an uncorrelated free-flight PDF that satisfies reciprocity over two-segment paths. Our $f_p$ function (Section 4.3) is equivalent to d’Eon’s uncorrelated free-flight PDF; however, we are able to show that the correlated and uncorrelated free-flight PDF are linked because of the underlying physical process of ensemble averaging. Reciprocity in our model is then simply a result of modeling the underlying physical process. In contrast, the uncorrelated free-flight PDF of d’Eon is a mathematical construct and is not a result of any particular property of the medium. The RTE derived by d’Eon ends up being only *weakly* reciprocal, i.e. the path throughput is not reciprocal for individual path segments, which complicates
rendering with bidirectional methods. Our path integral, on the other hand, is fully reciprocal, and also supports heterogeneity.

Jarabo et al. [66] also extend the model of Larsen et al. to include surfaces and statistical heterogeneity. Their RTE is general and allows for different correlated and uncorrelated source terms and free-flights. Although this RTE could be made reciprocal, Jarabo et al. do not explore the mathematical relationship between correlated and uncorrelated free-flight PDFs, and do not discuss reciprocity. In addition, although their RTE supports heterogeneity in theory, they do not develop a practical model and are only able to render homogeneous results. Finally, they do not provide a path integral formulation of their theory, which makes integration with bidirectional methods difficult.

In the field of neutron transport, Camminady et al. [21] extend the work of Larsen et al. [91] to support piecewise homogeneous media. They do so by combining a 1D free-flight PDF with the accumulated extinction across interfaces of the medium. Their model can be viewed as a special case of our heterogeneity model, which supports continuously varying heterogeneity. Unlike Camminady et al., we also provide a physical intuition of this heterogeneity model in the context of stochastic media and develop efficient and unbiased heterogeneous distance sampling methods for certain classes of media.

### 4.4 Modeling non-exponential attenuation

Now that we have a theory that can admit non-exponential behavior, the next remaining question is how we should obtain, represent, and design the four transport functions (4.27), and how we can physically interpret the corresponding light transport.
It is common to model surface appearance at multiple scales, e.g. by representing large-scale variation using explicit geometry and displacements, while modeling fine-scale roughness statistically using a BRDF. We propose to leverage our theory to provide a similar decomposition for volumes, where we model large-scale heterogeneous variation explicitly with spatially-varying macroscopic media properties \((\sigma(x), \rho_m(x))\), but can additionally account for scatterer correlations or unresolved fine-scale heterogeneity statistically by modifying the transport functions \((4.27)\). Inspired by phenomenological \([2, 114]\), data-driven \([3, 4, 98]\), and statistical microfacet \([15, 29]\) models for surface roughness, we can likewise obtain transmittance functions in different ways:

1. The artist-driven “phenomenological” way (Section 4.4.1), where we directly design a free-flight PDF/transmittance free-hand or with simple parametric models;

2. The “data-driven” physically based way (Section 4.4.2), where we instantiate a distribution of physical scatterers, and obtain the transmittance induced by these scatterers numerically (through sampling); and

3. The “statistical” physically based way (Section 4.4.3) that seeks analytic parametric models for these functions driven by some statistical description of the distribution of the physical medium scatterers (Section 4.4.4).

### 4.4.1 Transmittance via directly designed free-flight PDFs

A simple phenomenological approach is to directly prescribe the free-flight PDF (either Eq. \((4.27a)\) or \((4.27c)\)) to something other than an exponential. Table 4.1 lists the PDF pdf(x) and CDF cdf(x) of common statistical distributions. We can take any such distribution defined on the positive real line, and directly set e.g. \(fp(\tau) = \)
pdf(τ), we then have \( FF(τ) = 1 – cdf(τ) \), and the other functions follow from the relationships in Fig. 4.3.

We experimented with a variety of such distributions, including a step, linear ramp, sum of impulses, sum of exponentials, and Erlang-2. Fig. 4.5 shows renderings and the corresponding parametric transport curves. It would also be possible to allow artists to design custom curves using a familiar graph editor interface.

The sum of two exponentials is particularly useful as the two exponents allow separately controlling the falloff of light near the start and tail of the distribution. Wrenninge et al. [142] used this falloff for shadow rays to allow light to penetrate
Table 4.1: The probability density functions pdf(x), cumulative distribution function cdf(x), and characteristic functions φ(τ) for a variety of statistical distributions. Γ(α) and γ(s, x) are the complete and lower incomplete gamma functions, and δ(x) and H(x) are the Dirac delta and Heaviside step functions. When μ is used as a parameter it specifies the mean of the corresponding PDF. We omit the implicit values of 0 or 1 in PDFs and CDFs with bounded support. When hand-designing attenuation via free-flight distributions, pdf(τ) describes the free-flight distribution pp(τ), and transmittance is obtained via PF(τ) = 1 − cdf(τ); fp and FF are obtained via normalization and integration. When designing attenuation via 1/f noise, pdf(τn) describes the distribution of τn due to fBm micro-fluctuations, transmittance is obtained from ϕ(τn), and the free-flight distribution becomes −dϕ(τn)/dτ.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Parameters</th>
<th>pdf(x)</th>
<th>cdf(x)</th>
<th>φ(τ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta δµ</td>
<td>mean: µ &gt; 0</td>
<td>δ(x − µ)</td>
<td>H(x − µ)</td>
<td>e^{iτµ}</td>
</tr>
<tr>
<td>DeltaSum III(µ)</td>
<td>impulses:</td>
<td>1/n ∑_{i=1}^{n} δ(x − µi)</td>
<td>1/n ∑_{i=1}^{n} H(x − µi)</td>
<td>1/n ∑_{i=1}^{n} e^{iτµi}</td>
</tr>
<tr>
<td>Uniform U(a, b)</td>
<td>range: 0 &lt; a &lt; x &lt; b &lt; ∞</td>
<td>1/(b−a)</td>
<td>x−a)/(b−a)</td>
<td>erf[(x−a)/(2√v)]</td>
</tr>
<tr>
<td>Linear L(b)</td>
<td>range: 0 &lt; x &lt; b &lt; ∞</td>
<td>1/(b−a)</td>
<td>1−2x−b/2b^2</td>
<td>—</td>
</tr>
<tr>
<td>Normal N(µ, v)</td>
<td>mean: 0 &lt; µ &lt; ∞</td>
<td>(2πv)^−1/2 e^{−(x−µ)^2/2v^2}</td>
<td>1/2 [1 + erf((x−µ)/(√2v))]</td>
<td>e^{iτµ−v^2/2}</td>
</tr>
<tr>
<td>Gamma Γ(µ, α)</td>
<td>mean: 0 &lt; µ &lt; ∞</td>
<td>1/Γ(α) [x/α]^{α−1} e^{−x/α}</td>
<td>1/Γ(α) γ(α, xα/µ)</td>
<td>(1−irµ/α)^−α</td>
</tr>
<tr>
<td>Erlang-2 E(λ)</td>
<td>rate: λ &gt; 0</td>
<td>λ^2x e^{−λx}</td>
<td>e^{−λx} + λxe^{−λx}</td>
<td>(1−ir/λ)^−2</td>
</tr>
</tbody>
</table>

Deeper, approximating multiple scattering. Our theory allows us to incorporate this in a consistent way for arbitrary bounces and light transport algorithms.

Inspired by this idea, we also took a classical monopole diffusion profile [55, 77] and interpreted it as a free-flight distribution. The radial fall off of a classical monopole is proportional to an Erlang-2 distribution with rate parameter λ = √3[(1−α)/(1−gα)] dependent on albedo α and the average cosine g. Simulating single scattering from a point light in a homogeneous medium with this transmittance profile would produce results similar to an all-bounce monopole diffusion approximation. Applied as a transmittance profile in our theory, this lets light penetrate further into the medium at each bounce, and allows us to approximate multiple-scattered transport using fewer explicitly simulated bounces.
4.4.2 Data-driven transmittance via ensemble averaging

While directly designing transmittance functions in this top-down approach provides high-level artistic controls, it may not correspond to any physically realizable distribution of scattering particles. To obtain physically justifiable profiles, we can instead numerically approximate the ensemble-averaging process by generating stochastic realizations of heterogeneous media or discrete point distributions.

Without loss of generality, we will consider the function $F(t)$ knowing we can obtain the other three functions via integration, normalization or differentiation using the relationships in Fig. 4.3.

**Ensemble-averaging discrete media/particles**

One physically based approach would be to explicitly construct a discrete collection of scattering particles (Fig. 4.1a-d, top) and approximate the transmittance using a Monte Carlo ensemble average:

$$F(t) = \langle V(\mu(x,x_t)) \rangle \approx \frac{1}{N} \sum_{i=1}^{N} V(\mu_i(x,x_t)), \quad (4.29)$$

where $V(\mu)$ denotes the visibility within each random realization, and the ensemble average depends only on distance $t$ if the stochastic point process is statistically isotropic and homogeneous.

This ensemble average amounts to computing the probability of having no scattering particles over a distance $t$ along a random ray. If the positions of the particles are statistically independent, then this tabulation procedure will converge to the exponential distribution (see Fig. 4.1a,e), since it is computing the probability of no events occurring over a distance $t$ within a Poisson process. However, it is also possible to instantiate points with specially crafted spatial correlations that e.g. induce or inhibit clumping. Fig. 4.1e visualizes the transmittance curves obtained from 2D
discrete points sets with spectral power falloofs corresponding to “blue” (b), “pink” (c) and “red” (d) noise respectively. Data-driven curves like these are a “gold standard” in the sense that they can handle any distribution of scatterers we can explicitly construct. But they require expensive sampling and tabulation for every set of parameters, which quickly becomes intractable for large collections of particles in higher dimensions.

**Ensemble-averaging continuous densities**

We can forego instantiating discrete particles and instead model the spatial correlations via a heterogeneous micro-scale density field $\sigma_\mu(x)$ (Fig. 4.1a-d, bottom). Given a fixed realization of $\sigma_\mu(x)$, the realization-dependent transmittance can be written equivalently as:

$$\text{Tr}_\mu(x,x_t) = e^{-t \overline{\sigma_\mu(x,x_t)}}, \text{ where } \overline{\sigma_\mu(x,x_t)} = \frac{1}{t} \int_0^t \sigma_\mu(x,s) \, ds$$  \hspace{1cm} (4.30)

is the micro-scale density field $\sigma_\mu(x)$ averaged, or “blurred”, along a line segment of length $t$ between $x$ and $x_t$.

For a statistically homogeneous medium, the ensemble-average transmittance becomes:

$$FF(t) = \langle \text{Tr}_\mu(t) \rangle = \left\langle e^{-t \overline{\sigma_\mu(x,x_t)}} \right\rangle \approx \frac{1}{N} \sum_{i=1}^{N} e^{-t \overline{\sigma_{\mu_i}(x,x_t)}}.$$  \hspace{1cm} (4.31)

The Monte Carlo estimate is averaging transmittance through $\overline{\sigma_\mu(t)} = \overline{\sigma_\mu(x,x_t)}$ for a fixed $t$, but over random realizations $\mu_i$ of the medium. Fig. 4.6 shows random 1D transects $\sigma_\mu(t)$ (left) in a medium where $\sigma_\mu(x)$ is modeled using a single octave of Perlin noise [109], and transects (middle) of the corresponding “blurred” micro-scale density $\overline{\sigma_\mu(t)}$. 
Figure 4.6: Top left: 1D transects of a Perlin noise micro-scale density function $\sigma_{\mu}(t)$ for random rays and realizations, a single sample of which is highlighted in black. $\sigma_{\mu}$ is a random variable, the distribution of which, $pdf(\sigma_{\mu})$ is illustrated by the adjacent back histogram. Top right: The line-averaged micro-scale density $\overline{\sigma}_{\mu}(t)$ is likewise a random variable, but the distribution $pdf(\overline{\sigma}_{\mu}(t))$ now potentially depends on the length of the line segment $t$ as shown by the histograms for $t = 2.5$ (red) and $t = 22.5$ (green). Bottom: When the micro-scale density $\sigma_{\mu}(t)$ is modeled as $1/f^\beta$ fractal noise, the histograms of line-averaged micro-scale density $\overline{\sigma}_{\mu}(t)$ (thin transparent lines) take a closed form where the spectral exponent $\beta$ directly controls the variance $v_{\overline{\sigma}_{\mu}}(t)$ (thick opaque lines) as a function of distance $t$, where lower values of $\beta$ lead to more rapid decay.

While computing ensemble-averaged transmittance this way avoids the complexities of instantiating billions of discrete particles, it still makes parametric control cumbersome due to the need for tabulation.
4.4.3 Probabilistic ensemble-averaging

When performing ensemble averaging, \( \tau_\mu \) (or equivalently \( \tau_\mu \)) acts as a random variable with some probability density \( \text{pdf}(\tau_\mu \mid t) \) describing its variability for each distance \( t \). We visualize this as histograms on the right-hand side of Fig. 4.6.

**Longer-than-exponential tails.** This probabilistic view allows us to explain why in Fig. 4.1 the ensemble-averaged transmittance (red, pink) results in a longer tail than the exponential (black). Jensen’s inequality states that for a random variable \( X \) and a concave function \( f \): \( \langle f(X) \rangle \geq f(\langle X \rangle) \). Substituting the exponential for \( f \), and \( \tau_\mu(t) \) for \( X \), we should therefore expect:

\[
\langle e^{-t\tau_\mu(t)} \rangle \geq e^{-t\langle \tau_\mu(t) \rangle}. \tag{4.32}
\]

This will be an equality iff \( \tau_\mu(t) = \langle \tau_\mu(t) \rangle \) for all \( t \), which would mean the medium had no density fluctuations (was homogeneous) to begin with.

**Link to characteristic functions.** We can now write the ensemble average as the integral:

\[
\mathbf{F}(t) = \langle e^{-t\tau_\mu(t)} \rangle = \int_0^\infty e^{-t\tau_\mu} \text{pdf}(\tau_\mu \mid t) \, d\tau_\mu. \tag{4.33}
\]

This takes a form remarkably similar to the *characteristic function* (CF), which is the Fourier transform of a random variable’s PDF:

\[
\varphi_X(r) = \langle e^{irX} \rangle = \int_{\mathbb{R}} e^{irx} \text{pdf}_X(x) \, dx, \tag{4.34}
\]
where \( i = \sqrt{-1} \) is the imaginary constant. Comparing Eq. (4.34) to (4.33), we see that for a fixed \( t \) the ensemble-averaged transmittance is simply the CF of the random variable \( \sigma_\mu \) (by passing in \( t_i \) for \( r \)):\(^1\)

\[
\text{FF}(t) = \varphi_{\sigma_\mu(t)}(ti). \tag{4.35}
\]

This is a very powerful tool, because, as long as we can express the variation of \( \sigma_\mu \) (or \( \tau_\mu \)) using a distribution with a well-known CF (see Table 4.1), we can obtain closed-form expressions for the ensemble-averaged transmittance! Note that the parameters of the distribution pdf \( \sigma_\mu(t) \) may depend on the distance \( t \), as we saw in Fig. 4.6.

**Discussion.** So what should this distribution, and its dependence on \( t \) be? We could simply choose a convenient PDF and allow an artist to set its (potentially \( t \)-dependent) parameters by hand. While this would be fully parametric and analytic, it is unclear how (or whether) such a model would correspond to any micro-scale density fluctuations or correlations. Alternatively, we could ensure correspondence by fitting the distribution and its \( t \)-dependent parameters to tabulated data from realizations of \( \sigma_\mu(t) \), like in Fig. 4.6. Next, we will instead leverage recent work by Davis and colleagues (2011, 2014) to obtain a parametric, closed-form transmittance function whose parameters provide a physical interpretation of the underlying micro-scale extinction field as a fractal medium.

### 4.4.4 Closed-form average transmittance in fractal media

We will follow Davis et al. [34] and model the variability in \( \sigma_\mu(x) \) as a 3D fractal with 1D transects \( \sigma_\mu(x) \) characterized by \( 1/f^\beta \) fractal noise. Fig. 4.7 provides a visual

\(^1\) We could also express this using moment generating functions (MGFs), which are defined similarly to the CF, but without the imaginary constant.
Figure 4.7: $1/f^\beta$ noise (bottom) is characterized by a spectral exponent $\beta$ which determines the slope of its power spectrum on a log-log plot (top). Qualitatively, different values of $\beta$ lead to noises of different “roughnesses” (bottom), and integrating a noise with spectral exponent $\beta$ (e.g. white noise) produces another noise with spectral exponent $\beta + 2$ (e.g. red noise).

explanation: Qualitatively, $\beta$ determines how “rough” the noise will be (bottom), while quantitatively it dictates that the falloff of the function’s power spectrum will be $1/f^\beta$ (top). This directly controls the spatial correlations, where $\beta = 0$ means uncorrelated “white” noise, while $\beta > 0$ and $\beta < 0$ produce positive and negative correlations respectively.

We model $\sigma_\mu(x)$ statistically as a noise defined by: a mean $\mu$, a constant $C$ controlling its overall amplitude, and its spectral exponent $-1 \leq \beta \leq 1$ ranging from
“blue” ($\beta = -1$) to “pink” ($\beta = 1$). For convenience, we will combine these into a parameter vector $\psi = \{\mu_\sigma, C, \beta\}$. This allows us to write the ensemble-averaged transmittance (4.33) as:

$$\text{FF}(t \mid \psi) = \int_0^\infty e^{-t \sigma_\mu} \text{pdf}(\sigma_\mu \mid t, \psi) \, d\sigma_\mu,$$

where the PDF is now determined by the distance $t$ and parameters $\psi = \{\mu_\sigma, C, \beta\}$ defining the medium.

Given this fractal noise model, what can we say about $\text{pdf}(\sigma_\mu \mid t, \psi)$?

**Gaussian $1/f^\beta$ noise.** A key property of Gaussian $1/f^\beta$ noise is that it produces fractional Brownian motion (fBm) [97] with $\beta + 2$ via integration (arrows in Fig. 4.7).\(^2\) We therefore know that $\tau_\mu(t)$ will be a noise with $1 \leq \beta_\tau \leq 3$ and its distribution (as well as that of $\sigma_\mu$) will be a Gaussian. Davis et al. [34] formally derived the dependence of this Gaussian on the distance $t$ and medium parameters $\psi$:

$$\text{pdf}(\sigma_\mu \mid t, \psi) = N(\mu_\sigma, \nu_{\sigma_\mu}(t)), \text{ with } \nu_{\sigma_\mu}(t) = (C\mu_\sigma)^{\beta+1}t^{\beta-1}. \quad (4.37)$$

Fig. 4.6 visualizes random transects of $\sigma_\mu(t)$: the medium $\sigma_\mu$ averaged over a distance $t$. This figure shows the same information as the second row of Fig. 4.6, but this time for media modeled as fractal noise with $\beta = -1, 0, 1$. The superimposed analytic curves plot the standard deviation $\sqrt{\nu_{\sigma_\mu}(t)}$ from Eq. (4.37) as a function of $t$, matching the behavior of the random transects well.

---

\(^2\) $\beta$ is directly related to the Hurst parameter $H = (\beta - 1)/2$ more commonly use in the fractal literature [5] when describing fBm and the “persistence” parameter of fractal Perlin noise [40] used in graphics.
Using the characteristic function of the normal distribution (Table 4.1) with these parameters, we can obtain a closed-form expression for the ensemble-averaged transmittance via Eq. (4.35):

\[
FF_N(t \mid \psi) \approx e^{-\mu_v t + \nu \mu_v(t) t^2/2} = e^{-\mu_v t + (C \mu_v t)^{\beta+1}/2}
\] (4.38)

Unfortunately, this is only an approximation because the Gaussian is supported on the entire real line, but values of \(\sigma \mu < 0\) are non-physical. This model for noise is therefore only reasonable when \(\mu\) is set sufficiently high and \(v\) set sufficiently low so that negative extinction coefficients are unlikely to occur.

**Gamma-distributed 1/f^\beta noise.** To counteract the artifacts arising due to these negative intrusions, Davis et al. [34] proposed modeling the distribution of extinction values \(\sigma_v\) with the strictly non-negative Gamma distribution pdf \(pdf(\sigma_v \mid t, \psi) = \Gamma(\mu_v, \alpha(t))\), with parameters set to match the mean and variance of the Gaussian model. Combining Eq. (4.35) with Table 4.1 gives:

\[
FF_G(t \mid \psi) = \varphi_{G,t,\psi}(t) = \left(1 + \frac{\mu_v t}{\alpha(t)}\right)^{-\alpha(t)}
\] (4.39)

where

\[
\alpha(t) = \frac{\mu_v^2}{\nu \sigma_v(t)} = \frac{(\mu_v t)^{1-\beta}}{C^{1+\beta}}
\] (4.40)

is the Gamma model’s shape parameter that enforces a variance dictated by Eq. (4.37).

**Discussion.** Eq. (4.39) provides a simple 3-parameter model to produce non-exponential transmittance functions by specifying the mean extinction \(\mu\), the overall amplitude of variation \(C\), and the color/spectral falloff/correlation of the noise \(\beta\). Fig. 4.8 visualizes the four resulting transport functions for fixed \(C\) and varying \(\beta\) (top), and for fixed \(\beta = 1\) pink noise and varying \(C\) (bottom).
Figure 4.8: Ensemble-averaged transmittances (FF, PF) and free-flight PDFs (pp, fp) arising from Gamma-distributed fractal $1/f^\beta$ noise (4.39) with $\bar{\sigma} = 1/2$. Top: for a fixed $C = 3/2$ and different values of $\beta$; Bottom: for a fixed $\beta = 1$ and different values of $C$. Since $PF \propto fp$, we only show $fp$ in faded dashed lines in the middle plots. The inset on the bottom right shows the Gamma distribution controlling pdf$(\tau_\alpha)$ for pink noise.

As we would hope, both the Gaussian (4.38) and Gamma (4.39) models reduce to a simple exponential when variance $\nu \to 0$ (and hence $\alpha \to \infty$). This will happen if the medium is actually homogeneous ($C = 0$), or if we have white noise ($\beta = 0$), both of which correspond to independent scatterers. In the latter case of white noise, we obtain exponential falloff, but with a modified extinction coefficient.

“Pink” $1/f$ noise (when $\beta = 1$) is another interesting case because it has a “scale-invariant” property where the distribution pdf$(\tau_\alpha)$ in Eq. (4.38) and (4.39) no longer
depends on $t$ (see Fig. 4.6). This reduces the transmittance completely to the CF of $pdf(\mu_\mu)$:

$$
FF_{\text{pink}}(t) = \int_0^\infty e^{-t\mu_\mu} \, pdf(\mu_\mu) \, d\mu_\mu = \varphi_{\mu_\mu}(t).
$$

(4.41)

With the Gamma model, for instance, the variance that plugs into Eq. (4.39) reduces to $v_{\mu_\mu} = C^2 \mu_\mu^2$ when $\beta = 1$, and so $pdf(\mu_\mu) = \Gamma(\mu_\sigma, \alpha)$ where $\alpha = C^{-\beta-1}$ no longer depends on $t$.

Eq. (4.41) shows that ensemble averaging transmittance over a heterogeneous medium can be equivalently interpreted as averaging transmittance across realizations that are each homogeneous, but with a random extinction coefficient drawn from the distribution $pdf(\mu_\mu)$. This also gives a physical interpretation to other choices of non-exponential functions which are themselves characteristic functions of some distribution (e.g. the sum of exponentials [142] is the CF of a sum of two deltas $\Gamma(2)$): these all correspond to scale-invariant pink noise, but where $pdf(\mu_\mu)$ dictates the distribution of blurred extinction values the fractal takes on.

### 4.4.5 Combining macro- and micro- properties

We previously introduced our heterogeneity model (Section 4.3.2) and gave a physical interpretation of it in terms of “squeezing” and “stretching” the micro-density function by the macroscopic optical depth. However, with ensemble averaged transmittance from fractal media, we can only reason about the micro-density function in a statistical sense, and this physical interpretation changes slightly. Firstly, due to the self-similar nature of $1/f^\beta$ fractal noise, horizontal squeezing and stretching has no effect on the wavelength content and can be ignored. Instead, we can see from
Figure 4.9: In contrast to the deterministic case (Fig. 4.4), the statistical behavior of rescaled fractal noise depends on its power spectrum via Eq. (4.43). For instance, the standard deviation of blue noise ($\beta = -1$) does not depend on $\sigma$, while for pink noise ($\beta = 1$) it is proportional to $\sigma$. The colored intervals visualize how the standard deviation is affected by $\sigma(s)$ for $\beta = -1, 0, 1$.

Equations (4.38, 4.39) that scaling the distance $t$ by some constant $c$ is equivalent to scaling the mean $\mu_\sigma$ by the same constant:

$$\text{FF}(c \, t \mid \{\mu_\sigma, C, \beta \}) = \text{FF}(t \mid \{c \, \mu_\sigma, C, \beta \}). \quad (4.42)$$

This means that inserting the macroscopic optical depth adjusts the mean $\mu_\sigma$ of the micro-scale medium to locally match the density $\sigma$ at the macro-scale. Changing just the mean $\mu_\sigma$ of the noise, however, changes its standard deviation in a $\beta$-dependent way since:

$$v_{\sigma_\mu}(t \mid \{c \, \mu_\sigma, C, \beta \}) = c^{\beta+1} \cdot v_{\sigma_\mu}(t \mid \{\mu_\sigma, C, \beta \}). \quad (4.43)$$

This suggests that the way the density match is achieved has a different interpretation based on the color of the noise (Fig. 4.9). As in the deterministic case (Fig. 4.4), scaling the distance by a factor of $c$ in “pink” noise ($\beta = 1$) corresponds to scaling the micro-fluctuations vertically by the constant $c$, but as $\beta$ decreases the standard deviation is scaled less, until at “blue” noise ($\beta = -1$) it is not scaled at all ($c^0 = 1$), suggesting a vertical shift/translation of the noise instead.
Algorithm 1: Transport kernel evaluation and sampling for our method

```python
1 function \( \tilde{T}_{\text{base}}(\tau, \text{surfaceStart}, \text{surfaceEnd}) \)
2     |
3     if \( \text{surfaceStart} \land \neg \text{surfaceEnd} \) then return \( FF(\tau) \)
4     else if \( \neg \text{surfaceStart} \land \neg \text{surfaceEnd} \) then return \( pp(\tau) \)
5     else return \( PF(\tau) \)

function sample(\( x_i, \omega, \xi \))
6     surfaceStart ← \( x_i \in A \)
7     \( \tau^* \leftarrow \text{surfaceStart} \, ? \, \text{sampleFP}(\xi) : \text{samplePP}(\xi) \)
8     \( s \leftarrow \text{invertTau}(\tau^*) \)
9     \( z \leftarrow \text{raytrace}(x_i, \omega) \)
10    \( x_{i+1} \leftarrow x_i + \min(s, z) \cdot \omega \)
11    surfaceEnd ← \( s < z \)
12    if surfaceEnd then
13       pdf = \( 1 - (\text{surfaceStart} \, ? \, \text{sampleFP}_{\text{CDF}}(\tau^*) : \text{samplePP}_{\text{CDF}}(\tau^*)) \)
14    else
15       pdf = \( \sigma(x_{i+1}) \cdot (\text{surfaceStart} \, ? \, \text{sampleFP}_{\text{PDF}}(\tau^*) : \text{samplePP}_{\text{PDF}}(\tau^*)) \)
16    weight = \( \frac{\tilde{T}_{\text{base}}(\tau^*, \text{surfaceStart, surfaceEnd})}{pdf} \)
17    return \{weight, \( x_{i+1} \)\}
```

### 4.5 Implementation

In this section, we give a brief outline of the modifications that need to be made to an existing rendering algorithm to support our path integral. We outline the procedure for evaluation and sampling of \( \tilde{T} \) in Alg. 1. We assume that we are given functions \( FF \), \( pp \) and \( PF \) computed in a base medium (i.e. with macroscopic extinction of 1), as well as procedures \( \text{sampleFP}(\xi) \) and \( \text{samplePP}(\xi) \) to produce samples distributed proportional to \( fp \) and \( pp \), respectively. We also require functions to evaluate the PDF and CDF for a given sample.

Depending on how the transmittance was obtained, it may not be immediately obvious whether or not the equivalent base medium has an average extinction of 1 or not. While not a theoretical problem, deviations from 1 can be a nuisance in practice, as changing from an exponential to the non-exponential would change the apparent opacity of the medium, and the extinction coefficient would have to be modified to maintain the same overall appearance. We give an easy recipe to convert any transmittance into one that has an equivalent base medium with unit extinction.
Table 4.2: Sampling routines for the statistical distributions in Table 4.1.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Sampling</th>
<th>(x_i \propto \text{pdf}(x))</th>
<th>(\propto -d\varphi(ir)/dr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta (\delta_{\mu})</td>
<td>(x_\delta = \mu)</td>
<td>(-\frac{\ln \xi}{x_\delta})</td>
<td></td>
</tr>
<tr>
<td>DeltaSum III((n))</td>
<td>(x_{\text{III}} = \text{uniform from } {\mu_1, \ldots, \mu_n})</td>
<td>(-\frac{\ln \xi}{x_{\text{III}}})</td>
<td></td>
</tr>
<tr>
<td>Uniform (U(a, b))</td>
<td>(x_u = (b - a)\xi + a)</td>
<td>(-\frac{\ln \xi}{x_u})</td>
<td></td>
</tr>
<tr>
<td>Linear (L(b))</td>
<td>(x_l = b(1 - \sqrt{\xi}))</td>
<td>(-\frac{\ln \xi}{x_l})</td>
<td></td>
</tr>
<tr>
<td>Normal (N(\mu, \nu))</td>
<td>(x_n = \sqrt{-2\ln \xi_1} \cos(2\pi \xi_2))</td>
<td>(-\frac{\ln \xi_2}{x_n})</td>
<td></td>
</tr>
<tr>
<td>Gamma (\Gamma(\mu, \alpha))</td>
<td>Marsaglia’s method((\xi^{-1/\alpha - 1})\alpha/\mu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erlang-2 (E(\lambda))</td>
<td>(x_E = -\frac{1}{\lambda} \ln (\xi_1 \xi_2))</td>
<td>((\xi^{-1/2} - 1)\lambda)</td>
<td></td>
</tr>
</tbody>
</table>

First, we obtain the base extinction \(\bar{\sigma} = fp(0)\). Then, we construct new versions of the transmittance functions equivalent to a unit base extinction:

\[
\begin{align*}
\text{FF'}(\tau) &= \text{FF}\left(\frac{\tau}{\bar{\sigma}}\right) \quad (4.44) \\
fp'(\tau) &= \text{PF'}(\tau) = fp\left(\frac{\tau}{\bar{\sigma}}\right) \quad (4.45) \\
pp'(\tau) &= \text{pp}\left(\frac{\tau}{\bar{\sigma}}\right). \quad (4.46)
\end{align*}
\]

If the base extinction was already 1, then the transmittance functions are left unchanged.

Our sampling algorithm proceeds by generating a sample from \(fp\) or \(pp\), depending on whether \(x_i\) lies on a surface or not. Due to our heterogeneity model, we can interpret the generated sample as an optical depth \(\tau^*\) in the macroscopic medium and solve for the distance \(s\) along the ray such that \(\tau(x_i, x_i + s \cdot \omega) = \tau^*\). We do this analytically in homogeneous media, or numerically using regular tracking/raymarching in heterogeneous media.

The probability (density) of the generated vertex is obtained from the sample CDF (PDF) if \(s\) lies beyond (before) the distance to the nearest surface. The sample weight
is then simply $\tilde{T}/pdf$. If $fp$ and $pp$ are sampled perfectly (e.g. using formulas from Table 4.2), then this routine is an optimal distance sampling strategy.

### 4.5.1 Unbiased Delta Tracking

The sampling method in Alg. 1 relies on regular tracking (slow) or raymarching (biased) to sample distances. Methods based on delta tracking [27, 105, 116, 141] offer potentially better performance while remaining unbiased, but unfortunately rely on transmittance being an exponential (namely $Tr(a + b) = Tr(a)Tr(b)$). Fortunately however, our heterogeneity model allows us to use delta tracking for a subclass of non-exponential media.

For stochastic media characterized by $1/f$ (“pink”) fractal noise, Eq. (4.41) shows that the macroscopic transmittance is equivalent to ensemble averaging random homogeneous realizations. This means that an unbiased estimation of transmittance is possible by simply averaging scaled exponentials:

$$FF_{\text{pink}}(t) = \int_0^\infty e^{-t\tau_\mu} pdf(\tau_\mu) \, d\tau_\mu \approx \frac{1}{N} \sum_t e^{-t\overline{\tau}_\mu}, \quad (4.47)$$

where the scaling factors $\overline{\tau}_\mu$ are drawn $\propto pdf(\overline{\tau}_\mu)$. 

---

Figure 4.10: A heterogeneous medium rendered with regular tracking (left) and our unbiased delta tracking (right) for a Gamma distributed pink noise medium. Both methods are unbiased and have comparable noise levels, but delta tracking renders $23 \times$ faster.
Likewise, since our heterogeneity model inserts the macroscopic optical depth into the base transmittance, for a pink noise medium, we can globally multiply the heterogeneous medium by a constant factor and compute the classical transmittance, averaged over the distribution of scaling factors $\text{pdf}(\sigma_u)$. We can trivially turn this into a probabilistic process, in which we sample a scale factor $\propto \text{pdf}(\sigma_u)$, multiply the macroscopic density by this factor and compute the classical transmittance (or free-flight distance) through the resulting medium. This allows us to use any existing unbiased distance sampling algorithm, while obtaining a non-exponential result.

In Fig. 4.10, we demonstrate a cloud with Gamma distributed pink noise ($\text{pdf}(\sigma_u) \propto \Gamma(\alpha, \alpha)$, $\text{FF}(t) = (1 + t/\alpha)^{-\alpha}$ [35]) rendered with regular tracking as in Alg. 1 as well as classical delta tracking, for which the medium was scaled by a scaling factor sampled from a Gamma distribution at every bounce. Delta tracking converges to the same result as regular tracking, but $23 \times$ faster.

### 4.6 Results

We implemented our theory in two existing rendering systems, PBRTv3 [113] and Tungsten [9], following the outline in Section 4.5.
Figure 4.12: We show a cloud rendered with a traditional exponential transmittance (top) and with the non-exponential, long-tailed transmittance curve (bottom) of Davis et al. The non-exponential transmittance leads to both deeper light penetration as well as a softer appearance near the surface, allowing for a richer appearance.
Figure 4.13: We compare a dense homogeneous medium rendered with different non-exponential transmittances to an exponential reference. The mean free path of the non-exponential free-flight PDF was matched to that of the exponential. Despite matching the MFP, we are able to achieve a wide range of appearances.
In Fig. 4.5, we show a homogeneous medium rendered with five different designed transmittance curves and compare the results to an exponential medium. The non-exponential transmittances are simple parametric functions, such as linear or quadratic curves, but lead to a wide range of appearances that cannot be reproduced by the exponential. These curves were designed artificially and do not have a physical interpretation in terms of correlated scatterers, but their simplicity and ease of control makes them powerful tools for artistic control of the medium.

Fig. 4.11 shows a similar homogeneous medium with a linearly decreasing, non-exponential transmittance. The same scene is rendered with three different rendering algorithms—path tracing, light tracing and bidirectional path tracing—that were modified to support our theory. All three algorithms produce identical results, which supports the fact that our theory is reciprocal.

Fig. 4.12 shows a heterogeneous cloud, rendered with a traditional exponential transmittance, and a long-tailed non-exponential transmittance derived from the Davis and Mineev-Weinstein model. The long tail of the non-exponential transmittance allows light to penetrate deeper into the cloud, which leads to a brighter appearance near the bottom and a softer look near the surface, giving an overall richer appearance.

In Fig. 4.13, we compare different non-exponential transmittances to an exponential reference in a scene with subsurface scattering. The mean free path of the non-exponential free-flight PDF was matched to that of the exponential; however, despite having the same MFP, the non-exponential transmittance is able to achieve a wide range of appearances.

Finally, in Fig. 4.14 we compare a heterogeneous cloud rendered using a transmittance derived from fractal noise (bottom two rows) using the Davis and Mineev-Weinstein model with varying parameters ($\beta$, $C$). The fractal noise model allows a range of different appearances and mean free paths. For comparison, we also pro-
Figure 4.14: We compare a heterogeneous cloud rendered using a transmittance derived from fractal noise (bottom two rows) using the Davis and Mineev-Weinstein model with varying parameters ($\beta$, C). The fractal noise model allows a range of different appearances and mean free paths. For comparison, we also provide the same cloud rendered using an exponential with varying mean free paths (top row).
Monte Carlo Simulation

We additionally implemented our model in a simple Monte Carlo simulation (Fig. 4.15) for verification. We created a semi-infinite stochastic “red” noise medium and measured the mean penetration depth (after a fixed number of bounces) of photons emitted from a pencil beam at normal incidence. We compare the ground truth ensemble average against results of our path integral (“Ours”), the classical path integral and the model of Larsen et al. [91] (“GBE”) computed on a homogenized version of this medium. We additionally compare the ensemble average of a modified path integral in which photons “jump” to a different realization of the medium at every scattering event, conditioned on coinciding with a scatterer in the new realization (“Ours (model)”). The curves of our path integral match that of this modified path integral exactly, which verifies that our path integral computes precisely the ensemble average of this physical model. Additionally, the curves of our method are closer to ground truth than the classical exponential model or the model of Larsen.

Figure 4.15: We verify our model and compare it to ground truth in a Monte Carlo experiment in a semi-infinite medium. Please see Section 4.6 for details.
et al., which demonstrates that more sophisticated models of correlated scatterers and surfaces lead to more accurate results.

Although our method leads to more accurate results than previous work, we still observe some discrepancies between our model and ground truth. To better understand these differences, we implemented a modified version of our method that uses a free-flight PDF conditioned on the free-flight distance of the previous segment on the path (“Ours (two-bounce)”). Unlike the other methods, this model does not reset correlations at every interaction, but correlates free-flights of adjacent path segments. The disadvantage of this approach is that currently this conditioned free-flight PDF cannot be obtained analytically, and it must be tabulated in an expensive pre-process instead. Although not a practical approach today, this two-segment correlated method is closer to ground truth than our base model, which suggests that modeling correlations between more than two vertices can further improve accuracy.

4.7 Discussion

We conclude this chapter with a short summary of our contributions, and follow with a review of the limitations of our proposed model.

4.7.1 Contributions

In this chapter, we presented the following contributions:

- We introduced a new theory of volumetric light transport that allows for media with non-exponential free-flight distributions. Such distributions are the consequence of correlations between scatterers, which can arise from physical processes in e.g. the atmospheric sciences and neutron transport.
• We derived a new non-exponential path integral that approximates the general problem of ensemble averaged transport in stochastic media and supports surfaces and heterogeneity in a combined framework, while being reciprocal and practical to compute. Both our surface and heterogeneity models have simple physical interpretations and are easy to retrofit into existing volumetric rendering workflows.

• We presented a wide range of tools to leverage this newfound flexibility, ranging from simple parametric transmittance curves to powerful mathematical formalisms for Gaussian processes and fractal noise. The latter allowed us to obtain closed-form, ensemble averaged transmittances for scatterers distributed with different colors of noise.

Our theory can be implemented with only minor changes to existing rendering algorithms, and our results demonstrate that this allows for a rich range of volumetric appearances while still allowing for fully bidirectional transport. For the special case of pink noise media, we even adapted algorithms for unbiased distance sampling, which opens the door to efficient and unbiased rendering of non-exponential heterogeneous media.

4.7.2 Limitations

There are several limitations to our work that leave ample grounds for future research.

Our non-exponential path integral was derived via several assumptions and approximations to a general equation of correlated transport (Eq. (4.6)). Such approximations both restrict the types of correlations that can be expressed, as well incur approximation error when compared to the ground truth (Fig. 4.15). We focused primarily on correlations between the positions of scatterers, which impacts the trans-
mittance; correlations in other medium properties, such as the phase function, cannot be expressed in our framework. Additionally, we can only capture correlation along a single path segment. Correlations across multiple interactions in the medium are lost, and physically meaningful effects such as opposition surge are lost.

Our heterogeneity model also currently assumes that the scatterer correlations are the same throughout the medium (i.e. the medium is stationary). This means that we cannot handle multiple overlapping media with different transmittance functions for example, even though this could be a useful tool in practice. In addition, non-exponential media used for level-of-detail would benefit from spatially heterogeneous correlations, which our heterogeneity model cannot currently support.

In addition, our heterogeneity model currently lacks a model of scale. Some of the transmittance models we use are derived from probabilistic ensemble-averaging, either of concrete instantiations of scatterers or of fractal media. In either case, we fully homogenize the medium to derive a single transmittance function. However, if we use these transmittances together with heterogeneous densities in our framework, correlations on scales larger than the voxel size are accounted for twice: Explicitly, via the heterogeneous density; and implicitly, in the transmittance function. A better solution would be to indicate to the transmittance the scale at which homogenization occurs (i.e. the voxel size), so that only correlations below that scale are captured in the transmittance.

Although we can take advantage of unbiased distance sampling methods for certain media, it is unclear how to do this in the general case. More research is required to adapt efficient, unbiased distance sampling methods for all non-exponential heterogeneous media to our theory.

While our theory leads to a richer volumetric appearance space, navigating this space is still a challenge. Simply maintaining the same single-scattering mean-free path does not necessarily provide the best perceptual appearance match between a chosen non-exponential model and classical exponential transport. In fact, there are
two distinct mean-free paths in our framework (one for pp and one for fp), so this match is not even unique. Our experience suggests that the classical parameters (e.g. mean free-path) and the parameters of the correlations are not orthogonal, so adjusting them individually to reach a desired appearance is cumbersome. Reparametrizing the appearance space into perceptually uniform and orthogonal appearance dimensions, as has previously been investigated for color [124], glossy materials [140], and translucency [46, 147], would greatly facilitate future adoption of our model.

While Davis et al.’s (2011) 2-parameter fractal $1/t^\beta$ noise model theoretically allows changing the spectral exponent from $\beta = 1$ (pink) down to $\beta = -1$ (blue) noise, we found that, in our framework considering both media and surfaces, it produces non-physical free-flight PDFs and transmittances which go negative when $\beta \leq 0$ (see Fig. 4.16). We suspect this arises because Davis et al. operate exclusively on FF and make simplifying assumptions without enforcing the necessary constraints on FF’s first and second derivates to make our other three transport functions physically plausible. A more detailed investigation is warranted, but since $\beta \leq 0$ produces approximately exponential behavior, we have not found this reduction in the parameter range to be a concern.
4.7.3 Follow-up Work

Since the publication of this work, several related works have been published that extend or use some of the ideas proposed in this chapter.

Guo et al. [53] point out the limitations of the transmittance model of Davis et al. to domains for which \( \beta \leq 1 \), i.e. the range of stationary random fields [34]. Guo et al. use the theory of fractional Gaussian fields to derive a modified version of the model of Davis et al., such that \( v_{\tau_{\mu}}(t) \) in Eqs. (4.39) and (4.40) is replaced with a novel term that allows for values of \( \beta > 1 \). The resulting transmittance can represent long-range correlations that go beyond the pink noise (0 < \( \beta < 1 \)) regime.

Vicini et al. [136] use a non-exponential framework to approximate surface transport with a participating medium. They introduce a modified non-exponential theory that allows for a heterogeneously varying non-exponential transmittance, and thus varying correlations throughout the medium. They then introduce a new transmittance model that is a blend between classical exponential transmittance and a non-exponential transmittance with a linear falloff, the latter of which better approximates free-flight behavior near surfaces. Their combined model allows fitting a spatially varying blend ratio to best represent given geometry. Their final model is not reciprocal, and investigating reciprocal models that support spatially varying correlations would be a fruitful future research direction.
In Section 3.2.2, we briefly reviewed existing approaches to estimating light transport using photon mapping. Although the original photon mapping algorithm is based entirely around measuring the density of 0D photon points, more recent developments of line or beam-based samples [69, 72] have shown that the additional degree of freedom afforded by participating media can be exploited through the use of higher-dimensional 1D photon- and sensor samples. Detailed analysis by Krivánek et al. [85] shows that these higher-dimensional primitives can significantly outperform point-based photons as the density of the medium decreases.

In this chapter, we extend and generalize this idea to a framework of higher-dimensional photon and sensor samples. We will show that existing estimators using points (0D) and beams (1D) are just special cases of a more general theory of volumetric light transport simulation using higher-dimensional (nD) samples. Our theory operates by successively replacing propagation distance point samples with lines across multiple bounces to produce 2D (plane/quad) samples, 3D (volume/parallelepiped) samples, their camera path equivalents, and beyond. Fig. 5.1 (top) illustrates this process, and (bottom) demonstrates the potential of using the corresponding estimators for a simple volumetric lighting simulation. The additional dimensionality of our proposed sampling primitives not only decrease variance, but also bias, and a subset of our estimators correspond to unbiased photon estimators.
Figure 5.1: We generalize 0D photon points (a) and 1D beams (b) to produce progressively higher-dimensional nD samples such as 2D “photon planes” (c) and 3D “photon volumes” (d). We form these estimators by computing the limit process of “marching” or sweeping photons along preceding light path segments, which allows us to progressively reduce variance and bias. The motivational experiment in the bottom row uses these successive estimators (each shown vertically split at two sample counts) on a searchlight problem setup (left), confirming that higher-order nD samples have the potential to dramatically improve quality in volumetric light transport.

We begin this chapter with a review of the derivation of beam estimators from prior work in Section 5.1. We then generalize this derivation to obtain our proposed new nD primitives in Section 5.2, and show that prior estimators are just special cases of a more general theory that successively expands and estimates distance sampling steps. Extending the derivation from prior work becomes cumbersome for higher dimensions, and we propose a more general framework capable of deriving arbitrary photon surfaces in Section 5.3. We perform a detailed error analysis of our proposed estimators in Section 5.5, and conclude with the details of our implementation (Section 5.6) and an empirical evaluation of our method compared to prior work in Section 5.7.

5.1 Beams as a Limit Process

We continue the same notational convention from Section 3.2.2 and use $\bar{x}_i = x_0 \ldots x_i$ to denote a light subpath, and $\bar{y}_k = y_0 \ldots y_k$ to denote a sensor subpath. We assume that paths are sampled following the algorithm outlined in Section 3.2.1, where light
subpaths are constructed starting at the light by first sampling \(x_0\) and then sampling a sequence of directions and distances, which we group into \(\omega = \omega_1 \ldots \omega_l\) and \(t = t_1 \ldots t_l\) respectively.

This allows us to decompose the PDF of the subpath \(x\) beyond the first vertex as follows:

\[
p(x_1 \ldots x_l) = p(\omega_l) p(t_l), \quad \text{with} \quad p(\omega_l) = \prod_{i=1}^{l} p(\omega_i), \quad \text{and} \quad p(t_l) = \prod_{i=1}^{l} p(t_i). \quad (5.1)
\]

We similarly decompose the partial path throughput along subpath \(x\):

\[
g(x_l) = g(\omega_l) g(t_l), \quad \text{with} \quad g(\omega_l) = \prod_{i=1}^{l} g(\omega_i), \quad \text{and} \quad g(t_l) = \prod_{i=1}^{l} g(t_i), \quad (5.2)
\]

where

\[
g(\omega_i) = \Sigma(x_i) \rho(\omega_i) \quad \text{and} \quad g(t_i) = V(t_i) \text{Tr}(t_i) \quad (5.3)
\]

account for the directional and distance related throughput terms, respectively.

We also define a shorthand for the sample score of the light subpath:

\[
C(\omega_l) = \frac{g(\omega_l)}{p(\omega_l)}, \quad \text{and} \quad C(t_l) = \frac{g(t_l)}{p(t_l)}. \quad (5.4)
\]

We define all these quantities analogously for the sensor subpath \(y\), where \(s_i\) and \(\omega'_i\) denote the distance and direction arriving at sensor subpath vertex \(y_i\), to distinguish from the notation used for light subpaths.
5.1.1 Prior Density Estimators

With the notation established, we can now formalize the primary density estimators described in Section 3.2.2 in terms of approximations to the complete path throughput:

\[
\frac{g(z)}{p(z)} \approx C(w_1)C(t_{l-1})D^{l,k}C(s_{k-1})C(\tilde{w}_{k}),
\]

(5.5)

where \(D^{l,k}\) is the density estimator at the vertex pair \(x_1y_k\).

**Photon point–sensor point (oD \(\times\) oD, 3D blur).** The original volumetric density estimator connects a photon point to a sensor point using a 3D blur [76]:

\[
D^{l,k}_{P-P3D} = \frac{g(t_1)}{p(t_1)} \left\{ K_3(x_1,y_k)\rho^{l,k}_w \right\} \frac{g(s_k)}{p(s_k)},
\]

(5.6)

where \(K_3\) evaluates a 3D blur kernel at the vertices \(x_1y_k\), and \(\rho^{l,k}_w = \rho(-\omega_l,-\omega'_k)\) evaluates the scattering function at the connection point using the last photon and sensor subpath directions.

**Photon point–sensor beam (oD \(\times\) 1D, 2D blur).** In order to gather radiance along a sensor ray, a naive implementation would perform density estimation using \(D^{l,k}_{P-P3D}\) at regular intervals \(\Delta s\) in a raymarching procedure. The contribution of this estimator would be the sum of all such estimations:

\[
\sum_{i=0}^{\sum} \frac{g(t_1)}{p(t_1)} \left\{ K_3(x_1,y_k)\rho^{l,k}_w \right\} g(s^{(i)}_k)\Delta s,
\]

(5.7)

with \(s^{(i)}_k = i \cdot \Delta s\). Because of the deterministic marching procedure, the inverse sampling density \(p(s_k)^{-1}\) becomes \(\Delta s\). We now select a blurring kernel \(K_3(x_1,y_k) =\)
\[ \Delta s^{-1} \cdot K_2(x_l, y_k) \] which consist of non-overlapping cylindrical segments, the cross section of which is determined by \( K_2(x_l, y_k) \) (see figure below).

![Diagram](image)

The contribution then becomes:

\[
\sum_{i=0}^{\infty} \frac{g(t_l)}{p(t_l)} \left\{ \Delta s^{-1} K_2(x_l, y_k) \rho_{\omega}^{l,k} \right\} g(s_k) \Delta s.
\] (5.8)

Because the kernels are non-overlapping, the sum collapses to a single evaluation:

\[
\frac{g(t_l)}{p(t_l)} \left\{ K_2(x_l, y_k) \rho_{\omega}^{l,k} \right\} g(s_k^{(i^*)}).
\] (5.9)

where \( i^* \) corresponds to the interval containing the photon location \( x_l \), and \( \Delta s \) cancels. If we were to take infinitely many steps in the raymarching procedure, i.e. take the limit as \( \Delta s \to 0 \), we obtain the beam radiance estimator of Jarosz et al. [73]:

\[
D_{l,k}^{l,k} = \frac{g(t_l)}{p(t_l)} \left\{ K_2(x_l, y_k) \rho_{\omega}^{l,k} \right\} g(s_k^*),
\] (5.10)

where \( s_k^* \) corresponds to the location of \( x_l \) projected onto the sensor beam. Replacing the sensor point with a sensor beam both increases efficiency—the explicit summation is replaced with an analytic evaluation—and reduces bias, as the dimensionality of the blur is reduced by 1 to only a 2D blur.

**Photon beam–sensor beam (1D \times 1D, 2D blur).** This idea can be taken further by repeating the same procedure on the photon segment. Consider the sensor beam estimator from earlier, that now deterministically places photons on the last photon segment spaced \( \Delta t \) apart (see figure below).
The estimation becomes the sum of contributions from all the photons:

\[
\sum_{i=0}^{n} g(t_i) \Delta t \left\{ K_2(x_i, y_k) \rho_{\omega}^{l,k} \right\} g(s_i^*).
\] (5.11)

This corresponds to a Riemann sum evaluation. If we again take the limit of infinitely many photons with \(\Delta t \to 0\), the sum merges into an integral:

\[
\int_{t_{l,-}}^{t_{l,+}} g(t) \left\{ K_2(x, y_k) \rho_{\omega}^{l,k} \right\} g(s_k) \, dt,
\] (5.12)

where the integral bounds are defined by the intersection between the photon ray and the sensor beam, and variables \(s_k, x_i\) are functions of the integration variable \(t\).

If we swap the location of the blurring kernel, we obtain the sensor ray \(\times\) photon beam estimator of Jarosz et al. [70]:

\[
D_{B-B2D}^{l,k} = \int_{s_{k,-}}^{s_{k,+}} g(t) \left\{ K_2(x, y_k) \rho_{\omega}^{l,k} \right\} g(s) \, ds.
\] (5.13)

Similar to sensor beams, increasing the dimensionality improves efficiency even further compared to explicit raymarching. However, the dimensionality of the blur is not reduced.
Photon beam–sensor beam (1D×1D, 1D blur). To reduce the bias, we require a different raymarching procedure. Consider the blurring kernel $K_2(x_l, y_k) = u^{-1}K_1(x_l, y_k)$ that is oriented orthogonally to the sensor ray $\omega'_k$ and uniform in the plane of $\omega'_k$ and $\omega_l$. We now place photons regularly spaced $\Delta t$ apart and consider the estimate of the sensor ray:

$$\sum_{i=0} \Delta t \{ u^{-1}K_1(x_l, y_k)\rho^{1,k}_{\omega} \} g(s^t_k). \quad (5.14)$$

We now choose the blur width $u$ such that kernels of adjacent photons touch exactly when viewed from $\omega'_k$. This can be achieved by projecting the spacing between photons onto the uniform blur direction, yielding $u = \Delta t \| \omega_k \times \omega'_k \|$ (see figure below).

Because only one photon can overlap the sensor, the sum collapses and we obtain:

$$g(t^*_k) \Delta t \left\{ \frac{K_1(x_l, y_k)\rho^{1,k}_{\omega}}{\Delta t J^D_{B-B1D}} \right\} g(s^*_k), \quad (5.15)$$

where $J^D_{B-B1D} = \| \omega_l \times \omega'_k \|$ is the Jacobian for the 1D × 1D coupling. If we take the limit of infinitely many photons with $\Delta t \to 0$, we obtain a beam × beam estimator with a blur dimensionality of 1:
where $t_1^*, s_k^*$ are the distances to the beam-beam intersection along the photon and sensor segments. Compared to Eq. (5.13), the blur dimensionality is reduced by 1 to a 1D kernel.

### 5.2 nD photons as a limit process

In the previous section, we reviewed how previous beam estimators were the result of a limit process that placed estimators deterministically along a ray, and considered the limit of infinitely many such evaluations. In this section, we now generalize this concept to higher dimensions. Our key insight is that we can convert additional propagation distance steps to their expected values by unfolding the photon throughput prefix $C(T_1)$ and performing a limit process. We describe our approach by deriving novel two- and three-dimensional density estimators in detail; the same method can then be applied repeatedly to even higher dimensions.

**Photon plane–sensor beam (2D×1D, 1D blur).** We begin by inserting the B-B2D density estimator Eq. (5.13) into Eq. (5.5) to obtain

$$
\frac{g(z)}{p(z)} \approx C(\vec{w}_1) C(T_{l-1}) D_{B-B2D}^{l,k} C(\vec{s}_{k-1}) C(\vec{w}^*_k) = C(\vec{w}_1) C(T_{l-2}) \left\{ \frac{g(t_{l-1})}{p(t_{l-1})} D_{B-B2D}^{l,k} \right\} C(\vec{s}_{k-1}) C(\vec{w}^*_k).
$$

\[ (5.17) \]
The last step was achieved by assuming \( l \geq 2 \) and expanding \( C(\bar{t}_{l-1}) \) by one term. We will name the quantity inside the braces \( D_{\text{B-B2D}}^{l-1,k} \), which is a B-B2D estimator that performs one additional distance sampling step. Expanding this quantity yields

\[
D_{\text{B-B2D}}^{l-1,k} = \frac{g(t_{l-1})}{p(t_{l-1})} \int_{s_{k-1}}^{s_k} g(\tilde{t}_1) \left\{ K_2(x_l, \tilde{y}_k) \rho_{\omega}^{l,k} \right\} g(s) \, ds. \tag{5.18}
\]

The first term on the right-hand side is the result of distance sampling, which is used to obtain \( t_{l-1} \). We now replace this distance sampling step with a deterministic “beam marching” procedure (see figure below). Instead of sampling the location of a single beam, we place a series of beams at regular intervals along the ray \( x_{l-2} + \omega_{l-1} t_{l-1}^{(i)} \). We set the ray offset of each beam to \( t_{l-1}^{(i)} = i \Delta t \), where \( \Delta t \) is the step size.

We select a blurring kernel which is uniform along one dimension, \( K_2(x_l, y_k) = u^{-1} K_1(x_l, y_k) \), where \( u \) defines the uniform blur extent, and the direction of the uniform blurring is as in the figure below.

The contribution of this estimator then becomes a sum,

\[
\sum_{i=0}^{\infty} g(t_{l-1}^{(i)}) \Delta t \int_{s_{k-1}^{(i)}}^{s_k^{(i)}} g(\tilde{t}_1) \left\{ \frac{K_1(x_l, \tilde{y}_k)}{u} \rho_{\omega}^{l,k} \right\} g(s) \, ds. \tag{5.19}
\]

Because of the deterministic marching procedure, the inverse sampling density \( p(t_{l-1})^{-1} \) becomes \( \Delta t \). We now choose the uniform blur extent such that kernels...
of adjacent beams touch exactly, making \( s_{k+}^{(i)} = s_{k-}^{(i+1)} \). This is achieved with \( u = \Delta t \| \omega_{l-1} \times \omega_1 \| \). Substituting into Eq. (5.19) and rearranging yields

\[
\sum_{i=0}^{\infty} \int_{s_{k-}^{(i)}}^{s_{k+}^{(i+1)}} g(t_{l-1}^{(i)}) g(t_1) \Delta t \left\{ \frac{K_1(x_l, y_k)}{\Delta t} \rho_{l,k} \right\} g(s) \, ds, \tag{5.20}
\]

with \( J_{Q-B1D}^{l-1,1} = \| \omega_{l-1} \times \omega_1 \| \). The constant \( \Delta t \) can be moved into the braces and cancels. Taking the limit as \( \Delta t \to 0 \) merges the beams into a continuous photon plane with contribution

\[
D_{Q-B1D}^{l-1,k} = \int_{s_{k-}}^{s_{k+}} g(t_{l-1}) g(t_1) \left\{ \frac{K_1(x_l, y_k)}{\Delta t} \rho_{l,k} \right\} g(s) \, ds. \tag{5.21}
\]

**Photon plane–sensor beam (2Dx1D, oD blur).** In a similar fashion, we now insert the B-B1D estimator (Eq. (5.16)) into Eq. (5.5) and expand the distance throughput term to obtain the quantity

\[
D_{B-B1D}^{l-1,k} = \frac{g(t_{l-1})}{p(t_{l-1})} g(t_1^*) \left\{ \frac{K_1(x_l, y_k)}{\Delta t} \rho_{l,k} \right\} g(s^*). \tag{5.22}
\]

Again, we replace distance sampling along \( t_{l-1} \) with a deterministic beam marching procedure. We choose a uniform blurring kernel \( K_1(x_l, y_k) = u^{-1} \) with blur extent \( u \). The direction of the blur \( \bar{u} = (\omega_1 \times \omega'_k) / \| \omega_1 \times \omega'_k \| \) is oriented orthogonal to the last photon and sensor subpath directions (see figure on the next page).
The contribution then becomes

\[
\sum_{i=0}^{n} g(t_{l-1}^{(i)}) \Delta t g(t_{l}^{*(i)}) \left\{ \frac{K_{1}(x_{l}, y_{k})}{\int_{B-1D}^{l_{k}}} \rho_{l_{k}}^{1} \right\} g(s_{k}^{*(i)}) .
\]  

\[(5.23)\]

We choose \( u \) such that kernels of adjacent beams touch exactly when viewed from \( \omega'_{k} \). This can be achieved by projecting the spacing between beams onto the blur direction, yielding \( u = \Delta t |\vec{u} \cdot \omega_{l-1}| \). Since only one kernel overlaps the sensor ray, the summation disappears

\[
g(t_{l-1}^{*}) g(t_{l}^{*}) \Delta t \left\{ \frac{\rho_{l_{k}}^{1}}{\Delta t |\vec{u} \cdot \omega_{l-1}| \int_{B-1D}^{l_{k}}} \right\} g(s_{k}^{*}).
\]  

\[(5.24)\]

The constant \( \Delta t \) can be moved into the braces and cancels. Additionally, the term \( \|\omega_{l} \times \omega'_{k}\| \) occurs both in \( \int_{B-1D}^{l_{k}} \) and the denominator of \( \vec{u} \), and can be cancelled. Taking the limit and simplifying yields

\[
\mathcal{D}_{Q-BoD}^{l-1,k} = g(t_{l-1}^{*}) g(t_{l}^{*}) \left\{ \frac{\rho_{l_{k}}^{1}}{\int_{Q-BoD}^{l-1,k} \int_{1D}^{l_{k}}} \right\} g(s_{k}^{*}).
\]  

\[(5.25)\]

where \( \int_{Q-BoD}^{l-1,k} = |\omega_{l-1} \cdot (\omega_{l} \times \omega'_{k})| \) is the Jacobian for 2D×1D coupling with oD blur, yielding a continuous photon plane.
Photon volume–sensor beam (3D × 1D, 0D blur). We insert and expand the Q-B1D estimator 5.21 into Eq. (5.5) to obtain $P_{Q-B1D}^{1-2,k}$:

$$
g(t_{l-2}) \frac{1}{p(t_{l-2})} \int_{s_{k-}}^{s_{k+}} g(t_{l-1}) g(t_{l}) \left\{ \frac{K_1(x_{l}, y_k)}{\rho_{Q-B1D}^{1,k}} \right\} g(s) \, ds. \tag{5.26}
$$

We replace distance sampling along $t_{l-2}$ with deterministic “plane marching” and select a uniform blurring kernel $K_1(x_{l}, y_k) = u^{-1}$ with blur direction $\vec{u} = (\omega_{l-1} \times \omega_l)/\|\omega_{l-1} \times \omega_l\|$ normal to the plane (see figure below).

The contribution from all planes is

$$
\sum_{i=0}^{\infty} g(t_{l-2}^{(i)}) \Delta t \int_{s_{k-}}^{s_{k+}} g(t_{l-1}) g(t_{l}) \left\{ \frac{u^{-1}}{\rho_{Q-B1D}^{1,k}} \right\} g(s) \, ds. \tag{5.27}
$$

To ensure that adjacent planes touch exactly, we project the plane spacing onto the blur direction to obtain $u = \Delta t |\omega_{l-2} \cdot \vec{u}|$. Substituting into Eq. (5.27) and rearranging yields

$$
\sum_{i=0}^{\infty} \int_{s_{k-}}^{s_{k+}} g(t_{l-2}^{(i)}) g(t_{l-1}) g(t_{l}) \Delta t \left\{ \frac{|\omega_{l-2} \cdot \vec{u}|^{-1}}{\Delta t \rho_{Q-B1D}^{1,k}} \right\} g(s) \, ds.
$$
The term \(|\omega_{l-1} \times \omega_1|\) occurs both in the denominator of \(\vec{u}\) and the Jacobian \(J_{Q-BiD}^{l-1,1}\), and can be cancelled. Taking the limit and simplifying yields a continuous photon volume with contribution

\[
D_{B-BoD}^{l-2,k} = \int_{s_{l-2}}^{s_{l-1}} g(\tilde{t}_{l-2}) g(\tilde{t}_{l-1}) g(\tilde{t}_1) \left\{ \frac{\rho_{\omega}^{l,k}}{J_{C-BoD}^{l-2,1-1,1}} \right\} g(s) \, ds, \tag{5.28}
\]

where \(J_{C-BoD}^{l-2,1-1,1} = |\omega_{l-2} \cdot (\omega_{l-1} \times \omega_1)|\) is the Jacobian for 3D×1D coupling.

**Summary.** Our theory generalizes previous work by demonstrating how arbitrary numbers of distance sampling steps along photon paths can be replaced by expected value estimators, which ultimately results in higher-dimensional photon samples. While we explicitly derived new density estimators in 2D (photon planes, blurred and unblurred) as well as 3D (photon volumes), we can apply this process repeatedly to obtain estimators of arbitrary dimension.

While our derivations focused on photons, the exact same approach can also be applied to sensor paths, leading to novel sensor planes, sensor volumes and beyond. Additionally, these sensor primitives can be combined freely with photon planes, photon volumes and so forth. The expressions for the resulting density estimators only depend on the combined dimensionality of the photon, sensor sample, and blur kernel. It is worth noting that this combined dimensionality has to be at least 3 for density estimation to succeed. This explains why unbiased photon planes are possible, whereas e.g. Beam–Beam estimates require at least a 1D blur.

### 5.3 nD Photons as Solutions in an Extended Path Space

In the previous section, we derived novel photon estimators through a limit process of deterministically placed photons. These derivations relied on specific geometric constructions to correctly derive the Jacobian factors; while valid, this construction
is tedious and time-intensive. In this section, we give an alternative construction specifically for unbiased density estimators, which allows deriving new estimators with much less effort.

We begin by formulating the measurement equation as an integral over an extended path space $P^*$:

$$I = \int_{P^*} L_e(x_0) g(x_1, y_k) W_e(y_0) \, d\mu(x_1, y_k), \quad (5.29)$$

which integrates over all light and sensor subpaths, with the appropriate measure. We call this path space extended, because the path space is over-complete: For unbiased transport, we are interested in complete transport paths only, i.e. when $x_1 = y_k$ and the two connect into one full path. However, the space permits additional degrees of freedom where the subpaths do not connect. We have to remove these excess degrees of freedom by inserting a Dirac delta function into the path throughput:

$$g(x_1, y_k) = g(x_1) \delta(x_1 - y_k) g(y_k) \quad (5.30)$$

where $\delta(x_1 - y_k)$ selects only subpaths that connect to form a complete transport path. So far, this extended measurement equation is inconvenient to estimate: The manifold of subpaths whose endpoints connect has zero measure, and any straightforward Monte Carlo procedure for sampling subpaths in the extended path space would fail at producing any from this manifold.

To solve this, we perform two additional steps: First, we express the light and sensor subpaths $x(\xi)$ and $y(\xi)$ in terms of some abstract parametrization $\xi \in \Xi$ from parameter space $\Xi$.

This parametrization is again overcomplete; however, we now solve this issue by analytically integrating out three dimensions of the parametrization to reduce it back down to the same dimensionality as regular path space.
We can choose any three dimensions of the parametrization for this purpose, and we separate the parameter space \( \Xi = \Xi_a \cup \Xi_r \) into the set of \( \Xi_a \) analytically integrated dimensions, and the remaining dimensions \( \Xi_r \). The path throughput then becomes

\[
g(\bar{\xi}_r) = \int_{\Xi_a} g(\bar{x}_l(\bar{\xi})) \delta \left( x_l(\bar{\xi}) - y_k(\bar{\xi}) \right) g(y_k(\bar{\xi})) \left| \frac{\partial \left( x_l(\bar{\xi}) - y_k(\bar{\xi}) \right)}{\partial \bar{\xi}_a}(\bar{\xi}) \right| d\bar{\xi}_a. \tag{5.31}
\]

Compared to Eq. (5.30), all paths are now expressed as function of the parametrization. We also introduced an integral over analytical dimensions \( \Xi_a \), with a corresponding Jacobian determinant to account for the change of measure. Most importantly, the path throughput is now parametrized with the parameter vector \( \bar{\xi}_r \) (\( \bar{\xi} = \bar{\xi}_r \cup \bar{\xi}_a \)), which has the same dimensionality as the traditional path throughput.

If there is a unique \( \bar{\xi}_a^* \) for which \( \bar{x}_l(\bar{\xi}_a^*) = y_k(\bar{\xi}_a^*) \) (with \( \bar{\xi}_a^* = \bar{\xi}_a \cup \bar{\xi}_r \)), then the path throughput simplifies to

\[
g(\bar{\xi}_r) = g(\bar{x}_l(\bar{\xi}_a^*)) g(y_k(\bar{\xi}_a^*)) \left| \frac{\partial \left( x_l(\bar{\xi}) - y_k(\bar{\xi}) \right)}{\partial \bar{\xi}_a}(\bar{\xi}_a^*) \right|. \tag{5.32}
\]

This path throughput (and the corresponding path contribution) has all excess degrees of freedom removed and contains no integrals; it can be integrated over the remaining parameter space \( \Xi_r \) with standard Monte Carlo methods.

We will now proceed to make this definition concrete to rederive our previous estimators. Consider the path parametrization \( \bar{\xi} = \{x_0, \bar{\omega}_l, \bar{t}_l, y_0, \bar{\omega}'_k, \bar{s}_k\} \): This is the direction/distance parametrization we used in the previous section, with \( x_i = x_{i-1} + t_i \cdot \omega_i \). If we now pick \( \Xi_a = \{t_{l-1}, t_l, s_k\} \), i.e. select the two last distances on the light subpath, as well as the last distance on the sensor subpath, as our analytic dimensions, then we obtain the Jacobian

\[
\left| \frac{\partial \left( x_l(\bar{\xi}) - y_k(\bar{\xi}) \right)}{\partial \bar{\xi}_a}(\bar{\xi}_a^*) \right| = \left| \text{det}(\left[ \omega_{1-1}, \omega_1, \omega'_k \right]) \right| = \left| (\omega_{1-1} \times \omega_1) \cdot \omega'_k \right|. \tag{5.33}
\]
which is identical to the Jacobian $J_{Q-BoD}^{l-1,1,k}$ arising in the sensor beam $\times$ unbiased photon plane. Additionally, the parameters $\xi_{a}$ that connect both subpaths correspond exactly to the distances $\{t_{l-1}^{*}, t_{l}^{*}, s_{k}^{*}\}$ that arise from a sensor beam $\times$ photon plane intersection.

**Discussion.** We have rederived the photon plane estimator from the previous section through an alternative construction that involved temporarily raising the measurement equation to an extended path space, and subsequently integrating out the excess dimensions. We have done so without any raymarching arguments or geometrical constructions; instead, all terms arise naturally through Jacobians and constraint satisfaction. What is particularly exciting about this formulation compared to the derivation in the previous section is that it works for any choice of parametrization, and any choice of analytical dimensions $\xi_{a}$. For example, we could pick distance dimensions earlier on the light/sensor subpath; we could pick angular dimensions; we could even pick different parametrizations altogether. We call the family of estimators resulting from this construction photon surfaces, and they are explored in detail by Deng et al. [37].

### 5.4 Transmittance Estimators and “Long” vs. “Short” Beams

The photon primitives we have derived thus far correspond to beams, planes and volumes of (half-)infinite extent. These were previously characterized as “long” beams by Krivánek et al., in which a distance sampling step (to determine a subpath vertex location) is replaced by direct evaluation of transmittance. This reduces variance, but the infinite extent of these beams makes them challenging to incorporate into acceleration structures, and the appearance of a factor $g(t_{l})$ on its own requires the evaluation of a transmittance term at every estimation point, which can be costly in
heterogeneous media. Corresponding “short” beams also exist, in which the transmittance evaluation is replaced with an unbiased estimate \( \langle g(t_i) \rangle_T \) [71], which is equivalent to a constant step function that drops to zero after a randomly sampled propagation distance. Such a transmittance estimator results in a beam of finite extent and constant transmittance, and is thus much less costly to evaluate.

These operations have a tight link to transmittance estimation in neutron transport [126]. The original volumetric photon mapping approach (Eq. (5.6)) corresponds to estimating transmittance with a “collision” estimator (Fig. 5.2 (a)), where a photon scores a constant contribution if it falls within a finite interval (the blur kernel). The subsequent beam estimators replace a distance sampling step along the sensor and/or light path segment with a direct evaluation of transmittance. This corresponds to an “expected value” estimator (Fig. 5.2 (c)), so called because it returns the expected value of the propagation sampling process (the transmittance) directly [125]. The “short” beams correspond to estimating transmittance with a “track-length” estimator [125] from neutron transport (Fig. 5.2 (b)), which scores a constant value if the sampled distance lies beyond the estimation point.

Although we have consistently used expected value estimators to derive our photon planes and volumes, and transmittance evaluation can be replaced with a corresponding track-length estimator to form an equivalent short variant. For planes...
alone, this quadruples the number of estimators (long–long, short–long, long–short, short–short), depending on which segments use track-length estimation.

5.5 Theoretical Error Analysis

Since our theory predicts a growing number of possible estimators, we perform an error analysis to better understand their behavior compared to prior approaches. Our analysis is inspired by the one of Křivánek et al. [86], who examined variance in a single-scattering scenario with a point or beam estimator along a sensor ray and light ray. Since our theory allows unfolding the distance propagation term for additional segments, we generalize their analysis by considering a 2-scattering scenario involving three path segments and include bias in our analysis, leading to new insights.

5.5.1 Analysis Setup

We consider a canonical configuration for 2-scattering as depicted in Fig. 5.3. Since our theory only changes how the propagation distances are evaluated, we leave the directions of the light and sensor subpaths fixed, with the two directions of the photon path $\omega_1, \omega_2$ aligned with the $x$- and $y$-axes, and the sensor ray direction $\omega'_1$ set to the $z$-axis. The site of density estimation corresponds to the intersection of the sensor ray with the $xy$-plane. To simplify notation, we denote the distances to this point along each of the path segments as $x$, $y$, and $z$. Since the shape of the kernel is an arbitrary choice, we use a $d$-dimensional cube kernel of side-width $u$ as the resulting separability will simplify our analysis. This kernel will be the product of 1D kernels along the three path segments, producing a 1D step, 2D square, or 3D cube kernel.
Figure 5.3: Our canonical error analysis setup (top left) considers a double-scattering scenario where the two light path segments and the sensor segment are mutually orthogonal. We analyze the error for different choices of collision (C), track-length (T), or expected value (E) estimators for transmittance along these three directions. The images visualize results from a simulation using 100 photons for a grid of sensor rays spanning the xy-plane. The six images here correspond to the estimators that always use (E) along the sensor ray. The images are inverted to aid in visual comparison.

An abstract double-scattering estimator in this setup becomes

$$D_{x,y,z} \sim \langle A(x) \rangle \langle A(y) \rangle \langle A(z) \rangle u^{-d}$$  \hspace{1cm} (5.34)

where $u$ is the kernel width and $d$ the kernel dimension. We omit the directional scattering terms as they are fixed and influence the estimators by a shared constant factor. The three estimators $\langle A(x) \rangle, \langle A(y) \rangle, \langle A(z) \rangle$ determine the way transmittance is computed on the three segments, for which we consider three approaches:

$$\langle A(t) \rangle = \begin{cases} 
\langle \hat{T}r(t) \rangle_C & \text{collision estimator,} \\
\langle Tr(t) \rangle_T & \text{track-length estimator,} \\
\langle Tr(t) \rangle_E & \text{expected value estimator.}
\end{cases}$$  \hspace{1cm} (5.35)
Here \( \langle \text{Tr}(t) \rangle \) denotes a transmittance estimator, and \( \langle \hat{\text{Tr}}(t) \rangle \) an estimator of transmittance integrated over a kernel centered at \( t \):

\[
\hat{\text{Tr}}(t) \equiv \int_{t-u}^{t+u} \text{Tr}(t') \, dt' = \frac{e^{-\sigma(t-u)} - e^{-\sigma(t+u)}}{\sigma}.
\]  

(5.36)

### 5.5.2 Relation to Volumetric Density Estimators

The different combinations of these three estimators along the three segments yields \( 3^3 = 27 \) possible definitions of \( D_{x:y:z}^x \). This allows us to examine, for our canonical configuration, the beam and/or point estimators developed by prior work, and also the estimators we derived in Section 5.2. All estimators from prior work simulate 2-scattering by restricting at least one of the axes, typically \( x \), to a collision estimator, but allowing the other dimensions, \( y \) and \( z \), to be any of the three estimators in Eq. (5.35). For instance, using a collision estimator along all three axes corresponds to traditional volumetric photon mapping with a 3D kernel; using a collision estimator along \( x \), and track-length or expected value estimators along \( y \) and \( z \) result in the variety of short and long beam estimators from prior work. Our theory allows replacing the collision estimator along \( x \) with one of the other two estimators. For instance, using track-length estimators along \( x \) and \( y \), and an expected value estimator along \( z \) results in the “short–short” variant of estimator (Eq. (5.25)).

In Fig. 5.3 we visualize the results of a photon simulation in this canonical setup using several of these possibilities. The simulation generates 100 photon paths, and evaluates the estimators for a 2D grid of sensor rays parallel to the \( z \) axis.
5.5.3 Error Derivation

Given this generic setup, we wish to determine the relative root-mean-squared error (rRMSE) of Eq. (5.34):

$$\text{rRMSE}[D_{x,y,z}^*] = \sqrt{\mathbb{V}[D_{x,y,z}^*]} + \mathbb{B}^2[D_{x,y,z}^*]/D_E,$$  \hspace{1cm} (5.37)

where $E, \mathbb{V},$ and $\mathbb{B}^2$ are the mean, variance and squared bias of the estimator, and $D_E = \text{Tr}(x)\text{Tr}(y)\text{Tr}(z)$ is the unbiased reference value corresponding to using an expected value estimator along each axis.

Under our orthogonality assumption, the estimators along $x, y,$ and $z$ are statistically independent, so the expected value and variance of $D_{x,y,z}^*$ can be calculated from the first and second moments of $\langle \mathcal{A}(t) \rangle$ using the standard relations:

$$E[D_{x,y,z}^*] = u^{-d} E[\langle \mathcal{A}(x) \rangle] E[\langle \mathcal{A}(y) \rangle] E[\langle \mathcal{A}(z) \rangle],$$  \hspace{1cm} (5.38)

$$\mathbb{V}[D_{x,y,z}^*] = u^{-2d} (E[\langle \mathcal{A}(x) \rangle^2] E[\langle \mathcal{A}(y) \rangle^2] E[\langle \mathcal{A}(z) \rangle^2] -$$

$$E[\langle \mathcal{A}(x) \rangle^2 E[\langle \mathcal{A}(y) \rangle^2)] E[\langle \mathcal{A}(z) \rangle^2]),$$  \hspace{1cm} (5.39)

with squared bias:

$$\mathbb{B}^2[D_{x,y,z}^*] = (E[D_{x,y,z}^*] - D_E)^2.$$  \hspace{1cm} (5.40)
5.5 Theoretical Error Analysis

We can now analyze the estimators’ errors by plugging the first and second moments (Eq. (5.41–5.43)) into the definitions of mean, variance and squared bias (Eq. (5.38–

\[
\begin{align*}
\mathbb{E} \left[ \langle \hat{\text{Tr}}(t) \rangle_C \right] &= \hat{\text{Tr}}(t), & \mathbb{E} \left[ \langle \hat{\text{Tr}}(t) \rangle_C^2 \right] &= \hat{\text{Tr}}(t)/\sigma, \\
\mathbb{E} \left[ \langle \text{Tr}(t) \rangle_T \right] &= \text{Tr}(t), & \mathbb{E} \left[ \langle \text{Tr}(t) \rangle_T^2 \right] &= \text{Tr}(t), \\
\mathbb{E} \left[ \langle \text{Tr}(t) \rangle_E \right] &= \text{Tr}(t), & \mathbb{E} \left[ \langle \text{Tr}(t) \rangle_E^2 \right] &= \text{Tr}(t)^2.
\end{align*}
\]
5.5 THEORETICAL ERROR ANALYSIS

5.40), and finally into Eq. (5.37) for relative error. To reduce the number of possible estimators down from the available 27, we set the distances along each of the axes to be equal: \(x = y = z = 10 \text{ mfp}\), where \(\text{mfp} \equiv 1/\sigma\) is a mean free path. This leaves ten remaining unique possibilities, differing only by the number of times each type of estimator is used, and not their relative order. We denote these possibilities with the shorthand “C” for collision estimators, “T” for track-length, and “E” for expected value, and will express repetition of an estimator as an exponent. For instance, “C^2T” means two collision estimators and one track-length estimator. Fig. 5.4 compares the relative error of these ten estimators as we change the width \(u\) of the blur kernel.

**Impact of bias.** The intersection of the curves at \(u = 1\) is due to the fact that collision estimators have lower variance than track-length estimators for blur widths that are greater than 1 mfp [86]. By examining error instead of just variance, however, we see that as the kernel width for collision estimation continues to grow, the introduced bias dominates, and the error rapidly increases once again. This shows that collision estimators are not always better for large blur widths, but only if the blur width is within a sweet-spot range highlighted in Fig. 5.4 (approximately 1–9 mfps in our setup).

**Impact of expected value estimators.** Since the expected value estimator contributes no variance, employing it will always reduce variance compared to using the other estimators. This leads to three distinct groups of intersecting curves at \(u = 1\), which are determined by the number of times the expected value estimator is employed (decreasing/increasing our choice of \(x = y = z\) would furthermore bring these groups closer/further apart, respectively). The graphs also show that the expected value and track-length estimators always lead to a constant, finite variance, irrespective of the blur width, since they perform no blurring.
Impact of collision estimators for small widths. Unfortunately, the presence of just one collision estimator will lead to infinite variance as $u \to 0$. More interestingly, the rate at which the estimators shoot off to infinite variance differs based on the number of collision estimators used: the estimators with the same number of collision estimations (depicted in the same colors) share similar slopes. From a practical perspective this means that each additional replacement of a collision estimator with a track-length or expected value estimator will make variance asymptotically lower as the blur width diminishes. Current approaches use collision estimation for all but the two connecting segments, providing considerable opportunity for improvement especially for longer paths and small blur kernels.

5.5.5 Applicability to other Estimators

We only included estimators with the minimum required blur to make the number of possibilities manageable, but it would be trivial to also account for higher dimensions of blur by including track-length $\langle \hat{T}r \rangle_T$ and expected value estimators $\langle \hat{T}r \rangle_E$ of integrated transmittance in Eq. (5.35). We tried this for some common estimators, but since the total number of possible estimators for 2-scattering would grow to $3^5 = 243$ (or 56 if $x = y = z$), we have omitted these choices since we did not find them to provide further insights.

Since Eq. (5.34) is agnostic to which segments are on the sensor vs. light subpath, the results from our analysis also apply to other choices of sensor and light subpath lengths, as long as the total number of segments remains 3. For instance, the analysis is valid also for photon beam $\times$ sensor plane, or photon point $\times$ sensor volume.

While our analysis only considers up to double scattering, we anticipate that the insights gained here apply similarly to higher scattering orders. However, since it is not possible to create more than three mutually orthogonal directions in 3D, a
formal variance analysis for 3+ scattering would need to account for the statistical dependence of the estimators along the additional path segments.

### 5.5.6 Singularities

All of our estimators contain an inverse Jacobian in their contribution, but our canonical setup side-steps their influence by choosing orthogonal scattering directions. In practice, however, the relative directions of successive path segments could cause these inverse Jacobians to become singular. To understand the potential impact on variance, we note that the Jacobian for blurred photon planes is in fact identical to the Jacobian of the Beam×Beam (1D blur) estimator, so potential singularities should be of the same order. Further analysis is required to quantify the variance behavior of unblurred photon planes and photon volumes, which share a similar Jacobian. One important advantage of our estimators over prior work is that the Jacobians for blurred photon planes and photon volumes involve only the photon directions (and not directions on both the sensor and light subpaths), and could conceivably be cancelled using specially crafted importance sampling techniques [45].

### 5.6 Implementation

We validate our theoretical error analysis with implementations of several of the estimators predicted by our theory. Implementing all combinations of collision, track-length and expected value estimators across multiple photon path segments would be prohibitively time consuming, and instead we focus on a subset of 2D and 3D photon samples. Our most general implementation adds photon planes with track-length estimation to an open source ray tracing renderer [9]. We select short–short planes due to their finite extent, which makes them more amenable to efficient acceleration
Table 5.1: Absolute render times and speedup (in parantheses) of our performance optimizations applied to photon beams and planes across all seven of our test scenes, with respect to a baseline BVH implementation. Optimizations are enabled incrementally from left to right. Please see Section 5.6 for details.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Baseline BVH</th>
<th>Uniform Grid</th>
<th>Frustum Grid</th>
<th>Vis. Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B-BiD Q-BiD</td>
<td>Q-BiD Q-BiD</td>
<td>Q-BiD Q-BiD</td>
<td>Q-BiD Q-BiD</td>
</tr>
<tr>
<td>Bathroom</td>
<td>763s 32848 5201s</td>
<td>750s (1.02×) 1440s (2.28×) 3249s (1.60×)</td>
<td>1548 (4.96×) 13948 (2.36×) 2679s (1.94×)</td>
<td>5158 (10.10×)</td>
</tr>
<tr>
<td>Bedroom</td>
<td>905s 30128 4671s</td>
<td>973s (0.93×) 1403s (2.15×) 2797s (1.67×)</td>
<td>1808 (5.03×) 12878 (2.34×) 2312s (2.02×)</td>
<td>8078 (5.79×)</td>
</tr>
<tr>
<td>Kitchen</td>
<td>759s 23198 3145s</td>
<td>8148 (0.93×) 10958 (2.12×) 1886s (1.66×)</td>
<td>1658 (4.61×) 10598 (2.19×) 1635s (1.92×)</td>
<td>5668 (5.56×)</td>
</tr>
<tr>
<td>Living Room</td>
<td>614s 20658 3076s</td>
<td>5998 (1.03×) 876s (2.36×) 1840s (1.67×)</td>
<td>1448 (4.25×) 826s (2.50×) 1402s (2.19×)</td>
<td>5178 (5.94×)</td>
</tr>
<tr>
<td>Red Room</td>
<td>524s 2498s 3692s</td>
<td>356s (1.47×) 1022s (2.44×) 2080s (1.77×)</td>
<td>988 (5.32×) 10358 (2.41×) 1698s (2.17×)</td>
<td>6456 (5.73×)</td>
</tr>
<tr>
<td>Doorway</td>
<td>545s 23348 3190s</td>
<td>5498 (0.99×) 1479s (1.58×) 2351s (1.36×)</td>
<td>1948 (2.81×) 14338 (1.63×) 1803s (1.77×)</td>
<td>4468 (7.15×)</td>
</tr>
<tr>
<td>Staircase</td>
<td>491s 21868 2917s</td>
<td>568s (0.86×) 1019s (2.15×) 1819s (1.60×)</td>
<td>1348 (3.67×) 1902s (1.15×) 1631s (1.79×)</td>
<td>5068 (5.76×)</td>
</tr>
</tbody>
</table>
compared to their long counterparts. We also demonstrate a hybrid CPU-GPU implementation, which traces photon paths on the CPU and rasterizes the resulting photons using the GPU.

5.6.1 General Ray Tracing Renderer

Our most general implementation proceeds similar to a traditional two-stage photon mapping algorithm. In the photon shooting stage, we deposit photon points on surfaces and both beams and planes inside the medium. For every segment of the photon path that lies inside the medium, we insert a photon plane by sampling an additional scattering direction and propagation distance. If the segment originated on a surface, we additionally insert a photon beam.

Special care needs to be taken at the intersection of photon paths with solid objects. To ensure energy conservation, the length of the segments used for a short plane must always be the sampled propagation distance, not the distance to the nearest surface. Planes can therefore potentially extend beyond the surfaces they intersect. Visibility tests will prevent light leaks across surface boundaries.

Acceleration structure. Due to the size and distribution of the photon planes, ray-plane intersection tests can be challenging to accelerate using traditional bounding volume hierarchies. We instead use a uniform grid to store both planes and beams in our implementation. We additionally use a specialized frustum-aligned grid for intersection tests with primary rays, which allows for efficient culling. We provide detailed performance comparison of these optimizations applied to beams and planes in Table 5.1 with respect to a baseline BVH implementation. In order to provide a fair comparison, we use the same optimized data structures for both beams and planes in our implementation.
Visibility caching. In contrast to photon beams, photon planes require a visibility test along the last segment of the photon path for each evaluation. The performance impact of these visibility tests can be significant, and we employ a caching strategy to accelerate rendering with blurred photon planes. Conceptually, we blur the visibility term along the first segment of the plane by reusing visibility tests for samples in close proximity. For a first intersection point \( y^{(1)} \) (see figure below), we look up into the visibility cache consisting of regularly spaced bins of width \( u \) (the blur extent) along segment \( \omega_{l-1} \).

A cache miss results in a ray being traced along \( \omega_{l-1} \) toward \( y^{(1)} \), and the bin corresponding to \( t^{(1)}_{l-1} \) is populated with the distance to the nearest occluder. For a next intersection \( y^{(2)} \) with similar offset \( t^{(2)}_{l-1} \) along \( \omega_{l-1} \), the stored hit distance is reused to check for occlusion of \( y^{(2)} \), which avoids one ray tracing operation. Conversely, the cache lookup of an intersection \( y^{(3)} \) with too dissimilar hit distance \( t^{(3)}_{l-1} \) will fail, and a new ray is traced to populate the corresponding bin. To achieve a bounded memory footprint and avoid large up-front cost of the cache, we implement it as a per-thread hash map keyed by the index of the bin and the photon index. Performance savings depend on the blur extent of the plane, but are significant even on our test scenes with near imperceptible blur. We observe that the small additional
blur of the visibility introduced by the caching is insignificant, but we cannot use this caching strategy for unblurred planes if we wish for them to remain unbiased. For this reason, we only enable visibility caching for blurred photon planes. We provide concrete performance numbers of this optimization in Table 5.1 (right column).

**Control variates.** Computing the contribution of a blurred photon plane involves the evaluation of an integral along the segment of overlap on the sensor ray. Due to the presence of binary visibility in the integrand, this integral generally cannot be evaluated analytically, even in homogeneous media. Although the integral is straightforward to estimate with MC sampling, this process may introduce additional variance for long sensor segments. In practice, we notice that only a small percentage of visibility tests end up occluded, which motivates us to use a control variate [47] as a variance reduction technique. We provide full technical details of our approach in Appendix A.2. Although this procedure adds to the rendering cost (10%-15% in our scenes), for planes with large blur radii it can reduce variance significantly compared to naïve MC sampling. Its usefulness is reduced in our test scenes with small blur radii, but we still incorporate it to remain robust for all plane configurations.

Although our implementation uses a number of performance optimizations, we stress that these do not diminish the theoretical improvements of our estimators. Optimizing ray-photon intersections is not specific to our method, and indeed, uniform grids have been employed in previous work [86] to accelerate rendering with photon beams. Additionally, we use the same data structures for both beams and planes to provide a fair comparison; in fact, photon beams benefit significantly more from these optimizations than photon planes (Table 5.1).

Despite these optimization techniques, our implementation does not require parameter fine-tuning to achieve reliable performance. The only user-exposed parameter is the grid resolution, which does not noticeably affect performance over a reasonable range of values.
Table 5.2: Variance and effective speedup (in parantheses) of two of our estimators (Q-BoD and Q-B1D) and photon beams (B-B1D) on all seven of our test scenes. Please see Section 5.7 for details.

<table>
<thead>
<tr>
<th>Scene</th>
<th>B-B1D</th>
<th>Q-BoD</th>
<th>Q-B1D</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATHROOM</td>
<td>1.127</td>
<td>0.462</td>
<td>0.106</td>
</tr>
<tr>
<td>BEDROOM</td>
<td>0.901</td>
<td>0.156</td>
<td>0.071</td>
</tr>
<tr>
<td>KITCHEN</td>
<td>0.777</td>
<td>0.163</td>
<td>0.088</td>
</tr>
<tr>
<td>LIVING ROOM</td>
<td>0.182</td>
<td>0.032</td>
<td>0.011</td>
</tr>
<tr>
<td>RED ROOM</td>
<td>0.144</td>
<td>0.010</td>
<td>0.004</td>
</tr>
<tr>
<td>DOORWAY</td>
<td>0.171</td>
<td>0.060</td>
<td>0.008</td>
</tr>
<tr>
<td>STAIRCASE</td>
<td>0.276</td>
<td>0.073</td>
<td>0.020</td>
</tr>
</tbody>
</table>

5.6.2 Hybrid CPU-GPU Renderer

To generate the results in Fig. 5.1, we also implemented our approach using a hybrid CPU-GPU renderer specialized to the searchlight problem. Our implementation traces photon paths originating from a collimated beam through a semi-infinite homogeneous medium and rasterizes the resulting photon samples using the GPU. While this implementation only handles occlusion with the medium boundary, it demonstrates the relative performance of photon samples with up to 3 dimensions.

5.7 Results

We demonstrate our implementation on seven indoor scenes containing scattering media, and compare the effectiveness of two of our estimators (Q-B1D and Q-BoD) against photon beams. All render times were measured on a Linux cluster with 8-core 2.7GHz E5-2680 CPUs and 64GB RAM, using 16 threads.

We use our uniform and frustum-aligned grids to accelerate both beams and planes, with the grid resolution adjusted for best performance in each scene. In all
scenes, the grid resolution is relatively low (less than 50 cells on the largest dimension). Additionally, we make use of visibility caching and control variates for 1D blurred planes. We use the same blur radius for beams and blurred planes, which is set to cover roughly two pixels at the scene’s focal point. Since photon planes are more costly to evaluate than photon beams, we perform an equal-time comparison to include both the quality of the estimators and their cost in the comparison metric. Each estimator is run for 10 minutes on each scene.

Additionally, we compute the variance of each estimator as an objective comparison metric. We run 100 instances of each estimator with a different random seed, and compute the variance between all 100 renderings after five minutes of render time. For unblurred photon planes, the variance is simultaneously the error of the estimator, whereas photon beams and blurred photon planes contain an additional bias term that we do not compute. Since photon planes require a minimum of two or more scattering events in the medium, we exclude surface- and low-order medium scattering in our comparison and visualize only light paths with 2+ bounces in the medium. We stress that this is only for comparison reasons – our implementation can compute the full light transport in the scene.

We show renderings of four of our test scenes (Bathroom, Doorway, Kitchen, Living Room) at equal render time in Fig. 5.5. The top row/top left image shows the full light transport in the scene, whereas the remaining images demonstrate estimates of medium scattering obtained from each technique. For better comparison, we slightly boost the exposure of the medium-only images. We additionally show metrics across all seven of our test scenes in Table 5.2, both in terms of absolute variance and speedup (computed as the inverse of relative variance). The variance numbers were multiplied by a factor of $10^3$ for formatting reasons.

In all scenes, both of our plane estimators provide a substantial improvement in image quality at equal render time. Additionally, both of our plane estimators pro-
Figure 5.5: We show renderings of five volumetric scenes produced by three different estimators at equal render time. We show the full light transport and multiply scattered volumetric transport (exposure manually increased). Our estimators (photon planes, 0D and 1D blur) provide significant variance reduction and improved image quality compared to photon beams in all scenes.
Figure 5.6: Continuation of Fig. 5.5.
vide a significant decrease in variance, corresponding to a decrease in render time of $2.4 \times 40 \times$ compared to photon beams at equal quality.

Despite the fact that unblurred photon planes are unbiased, they provide less of an improvement than blurred photon planes in all of our scenes. We believe this to be for two reasons: Since we only measure variance, not error, the absence of bias in unblurred photon planes is not captured by our metrics. Additionally, we make use of visibility caching for blurred photon planes, which leads to a decrease in the number of visibility tests by a factor of 3 on average, significantly improving render times. The same technique cannot be applied to unblurred photon planes without introducing bias.

5.8 Discussion

We conclude this chapter with a short summary of our contributions, and follow with a review of the limitations of our proposed algorithm.

5.8.1 Contributions

In this chapter, we presented the following contributions:

- We presented a new theory of density estimation that generalizes prior work on point and beam samples to arbitrary dimensions. Our theory predicts an unbounded number of new and potentially unbiased estimators, both on the light- and sensor subpath.

- We present an extended path space that predicts an even larger class of unbiased photon surfaces. These can be interpreted as path sampling techniques, and trivially combined with other unbiased rendering algorithms with stan-
standard MIS techniques. The resulting wealth of estimators is explored in detail by Deng et al. [37].

• We extend prior variance analysis of these estimators to include their bias, and reveal that collision estimators only outperform track-length estimators in a narrow region of blur radii. We also show that in cases where photon beams provide error reduction over photon points, higher-dimensional photons compound this reduction even further.

• We demonstrate practical implementations of some of the estimators predicted by our theory, and show that they provide significant variance reduction and image quality improvement across a variety of scenes, both at equal sample count and equal render cost.

5.8.2 Limitations

The estimators we presented, as well as their implementation, suffer from a number of limitations that we would like to point out.

Heterogeneity. Although our “long” estimators support full heterogeneity, all of our implementations are based entirely on the “short” estimators, which are much easier to implement efficiently but do not perform well in heterogeneous media. Whenever track-length estimation is used on more than one path segment, the PDF used to construct the path no longer cancels with the transmittance and could introduce variance. Additional research would be required to handle short planes in heterogeneous media efficiently.

Visibility. One of the costliest aspects of our higher-dimensional estimators is the evaluation of visibility: Evaluating the contribution of an nD photon requires n – 1
visibility tests. We use a visibility cache in our implementation to significantly reduce
the number of visibility tests, but unfortunately this approach introduces additional
bias and cannot be applied to unbiased photon samples. It may be possible to apply
unbiased visibility caching techniques, such as unbiased deep shadow maps [71].

**Surface rendering.** Although our estimators improve efficiency for medium-to-
medium transport, surfaces do not currently benefit from our estimators. However, it
may be possible to leverage our framework to formulate novel and potentially unbi-
ased surface photons: While the intersection of photon beams with surfaces results in
photon points, surface intersections of photon planes induce line samples on surfaces,
and photon volumes result in photon polygons. Including such higher-dimensional
surface photons could lead to significant improvements in media-to-surface trans-
port, both in terms of variance and bias.
Earlier in this dissertation (Section 3.2.3), we reviewed a variety of MCMC based rendering algorithms in graphics. All of these existing methods fall into one of two camps: Methods that derive directly from MLT and operate entirely in path space, and methods that derive from PSSMLT and operate in primary sample space.

Perturbations and mutations that operate on paths directly have full control over the path geometry, and can employ sophisticated mechanisms tailored to specific lighting effects. On the other hand, the resulting algorithms are not robust when presented with material models or lighting effects not handled by any specialized perturbation, and come with considerable implementation complexity that makes widespread adoption difficult.

MCMC algorithms that operate in primary sample space on the other hand make use of an existing rendering algorithm that is treated as a “black box” mapping from random numbers to paths. On the one hand, this allows full use of the importance sampling techniques employed in existing renderers and results in increased robustness, on top of the ease of implementation afforded by the geometry of primary sample space. On the other hand, the Metropolis sampler in such methods has very little control or insight over how paths are constructed. This makes it impossible to handle difficult lighting effects with specialized perturbations. Additionally, specifically how the black box maps random numbers to paths can interact suboptimally with the Metropolis algorithm and lead to poor rendering performance.
In this chapter, we will introduce a new operation—reversible jumps—that can seamlessly transition between these two views. This operation allows combining both primary sample space- and path space perturbations in the same algorithm, and promises to leverage the best of both of these worlds. We will also introduce a new rendering algorithm, Reversible Jump MLT (RJMLT), as a practical demonstration of how this new operation can be used to improve existing algorithms.

We begin this chapter in Section 6.1 with a motivational example of why reversible jumps are useful. We then derive a more rigorous framework that can support such perturbations by importing the Reversible-Jump Markov Chain Monte Carlo [50] to graphics in Section 6.2. Our proposed operation relies on having inverses of rendering algorithms available. However, most rendering algorithms are not invertible, and we solve this problem in Section 6.3 by introducing an extended path space that matches the dimensionality of primary path space, and can keep track of the sampling information that is lost in regular path space. This theoretical construct is not practical to implement directly, and we show in Section 6.4 how it can be applied in practice without much overhead. With the framework established, we can compare our proposed method to the concurrent works of Pantaleoni [108] and Otsu et al. [106] in Section 6.5. Finally, we evaluate one potential application of our reversible jumps to an existing rendering algorithm in Section 6.6.

### 6.1 A Multiplexed Motivation

In this section, we begin with Multiplexed MLT as a motivational example that demonstrates why the “black box” view of the underlying rendering algorithm, and the resulting lack of control over the path, can be detrimental. We already alluded to this fact in Section 3.2.3.3 by reasoning about the structure that MMLT imposes on primary sample space.
We begin by making this structure explicit, and separating out the choice of sampling technique from the rest of primary sample space. The Markov Chain then operates in the product of a discrete and continuous space, $\mathcal{C}^k = \{0, \ldots, k + 1\} \times \mathbb{U}^k$ that we will call *multiplexed primary sample space*. This space can be decomposed into subspaces $\bigcup_{i=0}^{k+1} \mathcal{C}_i^k$, where $\mathcal{C}_i^k = \{i\} \times [0, 1]^\alpha_k$ is the subspace of the $i$-th sampling technique for paths of length $k$.

The efficiency gain of MMLT comes from its ability to explicitly select which sampling scheme $S_i$ should be used to turn primary sample space locations $\bar{u}$ into paths, which allows it to spend less time on ineffective sampling schemes. However, as we will see, MMLT has great difficulty in changing between different sampling schemes while maintaining local exploration.

The set of perturbations proposed in Multiplexed MLT will always perturb the position in primary sample space, and may additionally change the technique index. To simplify analysis, we will separate the two concerns and focus on a hypothetical TechniquePerturbation strategy, which keeps the position in primary sample space fixed and only attempts to transition to a different sampling technique.

That is, given the current state $\hat{u} = (i, \bar{u})$ in $\mathcal{C}_i^k$, the perturbation samples a technique index $j$ from the distribution $T(i \to j)$ and generates the proposal $\hat{v} = (j, \bar{u})$ in $\mathcal{C}_j^k$. The distribution $T(i \to j)$ is left unspecified, but for simplicity we assume it to be symmetric, i.e. $T(i \to j) = T(j \to i)$.

Since only the sampling technique is changed and not the random numbers, we would hope in the MMLT view that the acceptance probability will only depend on the ratio of MIS weights, i.e. on how well the proposed technique samples the current path compared to the current sampling technique. However, the actual acceptance probability is

$$r(\hat{u} \to \hat{v}) = \frac{C_j(\hat{u})T(j \to i)}{C_i(\hat{u})T(i \to j)},$$

(6.1)
Figure 6.1: Strategy changes in a multiplexed primary sample space such as that of MMLT generally lead to a large-scale change to the path geometry that causes the proposed path to be rejected with high probability. The RJMLT technique proposed in this chapter introduces efficient strategy perturbations that leave the path geometry intact.

where \( C_j(\bar{u}) = m_j(\bar{u})f_j(\bar{u})/p_j(\bar{u}) \) is the target function of MMLT. Expanding and simplifying yields

\[
 r(\hat{u} \to \hat{v}) = \frac{C_j(\bar{u})}{C_i(\bar{u})} = \frac{m_j(\bar{u}) C(S_j(\bar{u}))}{m_i(\bar{u}) C(S_i(\bar{u}))}. \tag{6.2}
\]

Eq. (6.2) exposes the main problem of this naïve perturbation: Even though the random number vector was not changed, the proposed state uses a different mapping to transform that vector into a light path. In general, \( S_i(\bar{u}) \neq S_j(\bar{u}) \), and it is likely that the proposed path (and therefore \( C(S_j(\bar{u})) \)) differs from the current path by a significant amount (Fig. 6.1). Such large changes are unlikely to be accepted, which impedes the Markov Chain’s ability to transition between different sampling strategies. Multiplexed MLT additionally couples technique changes with perturbations of \( \bar{u} \), but this does not address the problem observed in the naïve TECHNIQUEPERTURBATION: the algorithm suffers from low acceptance rates whenever a proposal changes strategies.
6.2 Reversible Jumps

Ideally, a technique perturbation would leave the current path unchanged while switching techniques. To do so, it must find a new point \( \bar{v} \) in primary sample space such that \( S_j(\bar{v}) = S_i(\bar{u}) \), and jump to \( \bar{v} \) as part of the perturbation.

Assume for now that sampling techniques \( S_i \) are well-behaved in the sense that they are diffeomorphisms, i.e. all \( S_i \) are smooth and possess a smooth inverse. Then, the desired perturbation is easily accomplished using the mapping \( \hat{v} = S_j^{-1}(S_i(\bar{u})) \). Here, we used \( S_j^{-1}(\bar{x}) \), which is an inverse sampling technique that transforms a light path into the random numbers that would produce it.

This perturbation is unusual, and it is not immediately clear how it should be used in MCMC to still lead to a correct result. To help analyze this problem, we introduce the Reversible Jump MCMC [51] framework (RJMCMC) to graphics. This methodology reasons about Markov chain transitions—or jumps—between different subspaces. It accomplishes this by modeling a jump with a deterministic mapping \( h_{ij} : \mathcal{U}^k \rightarrow \mathcal{U}^k \) that relates points from one space to the other; that is, \( \hat{v} = (j, h_{ij}(\bar{u})) \).

The acceptance probability for such a reversible jump is modified to incorporate the factor \( |J[h_{ij}]| \), which is the Jacobian determinant of \( h_{ij} \) to account for the density of the mapping.

In this framework, this proposed inverse sampling perturbation can be naturally expressed using the mapping \( h_{ij}(\bar{u}) = S_j^{-1}(S_i(\bar{u})) \). The Jacobian determinant of this mapping is

\[
|J[h_{ij}](\bar{u})| = |J[S_j^{-1} \circ S_i](\bar{u})| = |J[S_j^{-1}](S_i(\bar{u}))| \cdot |J[S_i](\bar{u})| = |J[S_j](\hat{v})|^{-1} \cdot |J[S_i](\bar{u})|, \tag{6.5}
\]
where the last step follows from the inverse function theorem. This allows us to write the Jacobian of the mapping in terms of the Jacobians of the sampling techniques\(^1\).

The Jacobian of a sampling technique is closely related to its PDF. Indeed, if \(S_i(\bar{u})\) is a diffeomorphism, then \(|J[S_i](\bar{u})| = p_i(\bar{u})^{-1}\) [80]. The acceptance probability of such a proposal is

\[
\begin{align*}
    r(\hat{u} \to \hat{v}) &= \frac{T(j \to i) C_j(\bar{v}) p_j(\bar{v})}{T(i \to j) C_i(\bar{u}) p_i(\bar{u})} \\
    &= \frac{T(j \to i) m_j(\bar{v}) f_j(\bar{v}) p_j(\bar{v})^{-1}}{T(i \to j) m_i(\bar{u}) f_i(\bar{u}) p_i(\bar{u})^{-1}} p_i(\bar{u}) \\
    &= \frac{T(j \to i) m_j(\bar{v}) f(S_j(\bar{v}))}{T(i \to j) m_i(\bar{u}) f(S_i(\bar{u}))} \\
    &= \frac{T(j \to i) m_j(\bar{v})}{T(i \to j) m_i(\bar{u})},
\end{align*}
\]

where the last cancellation was obtained with \(S_i(\bar{u}) = S_j(\bar{v})\), which holds by construction.

**Discussion.** Eq. (6.9) has several interesting properties. As desired, this acceptance ratio only depends on the relative MIS weight of the proposed and current technique. This allows the Markov Chain to easily transition to the optimal sampling techniques as it explores path space. Additionally, the path contribution function does not appear in the acceptance ratio, and expensive retracing of paths is not required. Notably, the remaining terms in the acceptance ratio can be derived from the current state before generating the proposal. This allows us to select an optimal proposal distribution with \(T(i \to j) = m_j(\bar{u})\). This will cancel all remaining terms and achieve an acceptance ratio of 1.

\(^1\) To side-step the subtlety that surface vertices of a light path \(x = x_1 \ldots x_n \in P\) reside on 2D subspaces of 3D, we assume that each vertex \(x_i\) can be locally parametrized using an orthonormal tangent frame. The concatenation of these local parameterizations then yields a local parameterization of the neighborhood of a light path, which facilitates reasoning about the densities of sampling strategies. For instance, \(|J[S_i](\bar{u})|\) refers to the Jacobian determinant of the \(i\)-th sampling strategy, which captures the change in volume when a small volume element in \(\mathcal{U}\) is mapped to \(\mathcal{P}\). This is purely a theoretical concern so these local parameterizations are not required during implementation.
We will refer to this perturbation as a \textit{reversible jump}, and the rendering method derived from it as \textit{Reversible Jump MLT}. The method proposed here can be summarized as:

1. Choose a proposal technique $j$ with probability $m_j(S_i(\bar{u}))$
2. Jump to proposal state $\hat{\nu} = (j, S_j^{-1}(S_i(\bar{u})))$
3. Always accept $\hat{\nu}$

\section*{6.3 Non-Invertible Sampling Techniques}

The perturbation introduced in the previous section assumed that sampling techniques are smooth and invertible, but this does not always hold in practice. There are two reasons for this: First, path space is usually \textit{over-parametrized} by sampling schemes in that the primary sample space has a much higher dimensionality than there are degrees of freedom on the path. This is problematic; not only are the sampling schemes not invertible, but the required Jacobians. Second, even if a sampling scheme preserves the same dimensionality, it is not necessarily injective, i.e. it may map several random number vectors to the same path, which introduces ambiguities when attempting to compute inverses.

We give three examples of when this commonly occurs in practical rendering algorithms:

\begin{enumerate}
\item \textbf{Discrete Decisions.}  A primary source of excess dimensionality in primary sample space are discrete decisions. For example, a rendering algorithm may need to decide whether to reflect or refract through an interface separating two dielectric materials. Say the reflection probability is $\alpha$, then a random number $u$ may be used to select one of the two cases by checking whether $u < \alpha$. Commonly, the random number is discarded after making this decision.
\end{enumerate}
Here, entire continuous intervals \((0, \alpha)\) and \([\alpha, 1]\) in primary sample space map onto the same path. If we wanted to invert \(u\), we could merely merely tell which interval \(u\) was originally in given the sampled direction, but inverting it is impossible. Even worse, primary sample space now has one additional degree of freedom than the direction, and the Jacobian determinant of the sampling scheme does not exist.

Technically speaking, this means that not even PSSMLT could be applied to this sampling scheme, as it relies on the Jacobian determinant in its construction (specifically, that the Jacobian determinant is the PDF). However, PSSMLT encounters such sampling schemes all the time in practice, and clearly this is merely a theoretical concern. We will later make explicit how the theory can be amended to avoid this issue.

2) Ambiguous Sampling Schemes. Even if excess degrees of freedom in primary sample space can be avoided, sampling schemes can still introduce ambiguity by mapping multiple random number vectors onto the same path. For example, a valid scheme for sampling a random direction that lies within a given solid angle is to randomly sample a point on a sphere of the same projected solid angle, and return the normalized direction to that point.

Although the point on the sphere has the same degrees of freedom as the direction—and thus, the mapping between the two has a Jacobian determinant—for any given direction, there are two possible points on the sphere that map to the same direction (front- and back-facing).

Mathematically speaking, this occurs whenever the mapping from random numbers to paths has a foldover, i.e. a point where the direction of the derivative reverses. At this point, the PDF of the sampling scheme is infinite; for this example, this happens on points on the silhouette of the sphere. Although foldovers can sometimes be avoided (e.g. by only sampling the front facing hemisphere), this cannot be guaranteed for all sampling schemes in practice.
3) Mixture Distributions. Lastly, both of these problematic cases can occur together. One common example is sampling a mixture of sampling techniques $S_a(u)$ and $S_b(u)$, so that the resulting PDF is $p_a(x)\alpha + p_b(x)(1 - \alpha)$. This arises in layered material models, for example.

A valid sampling scheme for this case is:

$$S_{\text{mixture}}(u) = \begin{cases} 
S_a(u_2, \ldots), & \text{if } u_1 < \alpha, \\
S_b(u_2, \ldots), & \text{otherwise.}
\end{cases} \quad (6.10)$$

Given a sampled point $x$, it is impossible to recover which of the two schemes produced it, meaning we do not even know which scheme to use to compute the inverse.

6.3.1 Extended Path Space

To address these issues, we will first introduce a conceptual construction that extends path space with auxiliary dimensions such that no information about the random numbers is lost during sampling, allowing paths to be inverted exactly. Afterwards, we will introduce probabilistic inverses as a practical solution to non-invertibility and reason about them within the RJMCMC framework.

Suppose that the set of all random number vectors that map to a given path $\bar{x}$ is given by $M_{\bar{x}} = \{\bar{u} \mid S_j(\bar{u}) = \bar{x}\}$. In principle, we should be able to augment the inverse $S_j^{-1}$ with an additional parameter $\bar{\gamma} \in [0, 1]^m$, that disambiguates the inverse so each pair of inputs $S_j^{-1}(\bar{x}, \bar{\gamma})$ maps to a single entry of $M_{\bar{x}}$. The role of $\bar{\gamma}$ will be to encode the “extra” dimensions in $\bar{u}$ that do not directly sample the path. If such a map can be constructed (assuming a sufficiently large value of $m$), then we can perform reversible jumps with the following modifications: sampling techniques generate points on an extended path space $\mathcal{P} \times [0, 1]^m$, and $S_i(\bar{u})$ computes the pair
where the step from Eq. (6) to primary sample space. We sidestep this issue by ensuring all mappings are bijective (right); we extend the path space parameterizing its points by an additional parameter $\gamma$ produced by $S_j(\bar{u})$.

$(\bar{x}, \bar{y})$, where $\bar{x}$ is the sampled path, and $\bar{y}$ is the auxiliary input that would produce $\bar{u}$ when used in the inverse $S_i^{-1}(\bar{x}, \bar{y})$; see Fig. 6.2 for an illustration.

Using the extended path space, no information is lost when transitioning from or to primary sample space, enabling the use of RJMCMC to derive the acceptance probability

$$r(\bar{u} \to \bar{v}) = \frac{C_j(\bar{v}) T(j \to i) |J[S_i](\bar{u})|}{C_i(\bar{u}) T(i \to j) |J[S_j](\bar{v})|} \quad (6.11)$$

$$= \frac{C_j(\bar{v}) T(j \to i) |J[S_i^{-1}](\bar{x}, \bar{y})|^{-1}}{C_i(\bar{u}) T(i \to j) |J[S_j^{-1}](\bar{y}, \bar{y})|^{-1}} \quad (6.12)$$

$$= \frac{C_j(\bar{v}) T(j \to i) |J[S_i^{-1}](\bar{y}, \bar{y})|}{C_i(\bar{u}) T(i \to j) |J[S_j^{-1}](\bar{x}, \bar{y})|} \quad (6.13)$$

where the step from Eq. (6.11) to (6.12) follows from the inverse function theorem.

The mapping from primary sample space to the auxiliary variables $\bar{y}$ is not arbitrary: The reciprocal Jacobian determinant of this mapping must match the PDF returned by the path sampling scheme in order for the PSSMLT construction to hold. Luckily, as long as the PDF exists, we know that the corresponding mapping must
exist also (as the PDF is defined by the Jacobian of the mapping). In any existing sampling scheme, we can therefore simply think of the mapping as being implicit.

For the example of a discrete decision, the mapping could look like

\[
\gamma = \begin{cases} 
\frac{u}{\alpha} & \text{if } u < \alpha \\
\frac{u - \alpha}{1 - \alpha} & \text{else.}
\end{cases}
\]

(6.14)

The reciprocal Jacobian determinant of this mapping corresponds to \(\alpha\) or \((1 - \alpha)\) depending on the outcome, which indeed matches the probability of the outcome.

**Probabilistic inverses.** This construction provides a viable way of supporting non-invertible mappings, but it would not be practical to store and propagate the auxiliary dimensions throughout the entire rendering system. Instead, we rely on a much lighter-weight solution to resolve ambiguities during path sampling: Whenever multiple inverses are available, we simply randomly select one. This is realized by combining strategy change perturbations with an additional step that samples \(\bar{\gamma} \in [0, 1]^m\) from a uniform distribution. Since the auxiliary vector \(\bar{\gamma}\) is re-sampled as part of every strategy perturbation, it is no longer part of the Markov Chain’s state, and does not need to be computed and stored explicitly—it manifests simply as additional random decisions when inverting a path.

### 6.4 Inverses in Practice

In the previous section, we derived a general framework for handling path sampling methods that are not strictly invertible, and reconciled these ideas with the RJMCMC framework. In this section, we will now discuss inverses in more concrete terms and describe a few simple building blocks that allow correct inversion of many sampling
methods used in light transport simulations. We will also discuss how to compute
the required Jacobians in more detail.

As previously discussed in Section 3.2.1, paths are almost always sampled incre-
mentally from a random walk in practice. This corresponds to chaining a series of q
low-dimensional sampling techniques, and we can write

\[ S_i(\bar{u}) = (g_1(u_1), g_2(x_1, u_2), \ldots, g_q(x_{q-1}, u_q)) , \]  

where \( \bar{x}_l = x_1 \ldots x_l \) is the path up to vertex \( x_l \), and \( g_l(x_{l-1}, u_l) \) is the l-th sampling
technique along the path, using a subset of the random numbers in \( \bar{u} \). Inverting
\( S_i \) then reduces to inverting the individual techniques \( g_l \), in an analogous “inverse
random walk”. The Jacobian determinant required by Equation (6.13) turns into a
product of determinants for each step, i.e.,

\[ |J[S_i^{-1}](\bar{x}, \bar{\gamma})| = \prod_{l=1}^{q} |J[g_l^{-1}](x_l, \gamma_l)| . \]

In order to maintain an optimal acceptance ratio and for Eq. (6.9) to hold, we wish for
the individual Jacobians \( |J[g_l^{-1}](x_l, \gamma_l)| \) to be equal to the PDF of the corresponding
mapping \( g_l \). In the following, we will focus on a single sampling technique \( g \), and
drop the subscripts for ease of notation.

**Inversion method.** Mappings based on the inversion method (Section 3.1.4.1) form
the basic building blocks of many sampling techniques and are easily handled. Such
techniques draw samples from a probability distribution \( p \) by mapping uniform
variates through the inverse CDF \( P^{-1} \). These mappings are invertible by construction,
and to support them in our system we simply require an additional implementation
of the CDF \( P \), which serves as a (non-probabilistic) inverse. The Jacobian determinant
\( |J[g^{-1}](x, \gamma)| \) of this inverse is equal to the PDF \( p \) of the sampling technique.
In the remainder of this section, we turn to the sources of non-invertibility discussed in Section 6.3.

6.4.1 Ambiguous Intervals

Ambiguous interval arose whenever a dimension $u_i$ in primary sample space could take on any value on an interval $[a, b]$ without changing the generated path. Constructing a probabilistic inverse for this case is fortunately easy: we generate a uniform variate $\gamma \in [0, 1]$ and set $u_i = g^{-1}(\gamma)$, where

$$g^{-1}(\gamma) = a + \gamma \cdot (b - a) \quad \text{and} \quad J[g^{-1}](\gamma) = b - a.$$  \hspace{1cm} (6.16)

For entirely unused dimensions of primary sample space this reduces to $g^{-1}(\gamma) = \gamma$ with a unit Jacobian determinant; that is, unused dimensions are simply uniformly re-sampled during inversion and do not influence the acceptance probability.

6.4.2 Mixtures of Sampling Techniques

Suppose that $g$ consists of a combination of sampling techniques $g_1, \ldots, g_n$ selected at random, where technique $g_i$ is chosen with probability $\alpha_i$. We assume that the technique index $t$ is chosen by the primary sample $u_1$ such that $\alpha_1 + \ldots + \alpha_{t-1} \leq u_1 < \alpha_1 + \ldots + \alpha_t$.

We now propose a probabilistic inverse that resembles this sampling procedure. First, we randomly select a technique index $t$ from a (yet undisclosed) discrete distribution $T(t)$. We then invert the sample assuming that it was generated by the $t$-th technique. This disambiguates both which interval the variate $u_1$ falls into, as well as
which mapping $g_t^{-1}$ should be used. For a fixed $t$, the resulting inverse and Jacobian determinant are then

$$g^{-1}(x, \gamma) = (\alpha_1 + \ldots + \alpha_{t-1} + \gamma_1 \cdot \alpha_t, \quad g_t^{-1}(x, \gamma_2, \ldots)), \quad (6.17)$$

$$|J[g^{-1}](\bar{x}, \gamma)| = \alpha_t \cdot |J[g_t^{-1}](\bar{x}, \gamma_2, \ldots)|. \quad (6.18)$$

This presence of a technique index $t$ extends the proposal state generated by RJMLT: In addition to selecting the $j$-th technique of BDPT as MMLT does, the strategy perturbation also selects which of the $n$ sampling techniques should be used to invert $g$, which yields a slightly modified acceptance probability

$$r((\hat{u}, t_0) \to (\hat{v}, t_\Psi)) = r(\hat{u} \to \hat{v}) \frac{T(t_0)}{T(t_\Psi)}. \quad (6.19)$$

The last step is to pick a concrete distribution $T$. Any distribution that samples strategy $t$ with nonzero probability if it could potentially have produced $x$ is in principle admissible. We use

$$T(t) = \frac{\alpha_t \cdot |J[g_t^{-1}](\bar{x}, \gamma)|}{\sum_{s=1}^n \alpha_s \cdot |J[g_s^{-1}](\bar{x}, \gamma)|}, \quad (6.20)$$

which cancels out the Jacobian (6.18) in the acceptance ratio, and results in an acceptance rate of 1.$^2$

**Final algorithm.** For completeness, we now state the outline of our complete strategy perturbation:

1. Choose a proposal technique $j$ with probability $m_j(S_t(\bar{u}))$

2. Jump to proposal state $\hat{\Psi} = (j, S_j^{-1}(S_t(\bar{u})))$

$^2$ $T$ has an intuitive interpretation: If $g_t$ is itself not a nested mixture of techniques, then $T(t)$ is simply the discrete probability that $\bar{x}$ was generated by the $t$-th technique.
• If there are ambiguous intervals for elements of \( \hat{v} \), sample them uniformly (Equation 6.16)

• If \( S_j \) uses a mixture of sampling techniques, select one randomly according to \( T \) (Equation 6.20)

3. Always accept \( \hat{v} \).

There is one small caveat to step (3): in rare cases, it might be impossible to invert a path due to numerical error. We detect such cases by checking if the path cannot be sampled by the method that we wish to invert (for example, if a direction lies in the wrong hemisphere), and reject proposals in this case.

### 6.5 Comparison to Concurrent Work

Concurrent to our work, Pantaleoni [108] and Otsu et al. [106] also introduce the idea of using inverses in the context of PSSMLT. Now that our framework is established, we can compare between these three works in more detail.

Pantaleoni [108] bridge path space and primary sample space with a parametric family of charts and their inverses, and design novel Metropolis-based rendering algorithms based on bidirectional path tracing and density estimation. Otsu et al. [106] employ invertible mappings to transform samples between (multiplexed) primary-sample space and path space. This allows them to combine perturbation strategies from both spaces.

In contrast to these two works, we recognize the theoretical problem of the mismatched dimensionality between primary sample space and path space and the corresponding lack of Jacobians in mappings between the two. We perform a thorough treatment of this issue by introducing both Reversible Jump MCMC and an extended path space. This gives both PSSMLT and our RJMLT a solid theoretical foundation, and simultaneously informs how to implement inverses correctly. While we give ex-
plicit recipes for inverting common sampling techniques used in practice, our theory can also be used to derive inverses for sampling techniques with corner cases we did not treat explicitly in this chapter. This rigorous treatment is missing from the two concurrent works.

The work of Otsu et al. [106] focuses on combining MMLT with path space perturbations, while we instead show an application that allows optimal changes of sampling techniques in MMLT. Our techniques can be seen as orthogonal, and their combination would be worth investigating in future work. The work of Pantaleoni [108] also introduces a technique change to MMLT. The underlying ideas are the same as our proposed perturbation, and are simply derived using different mathematical frameworks. In contrast to Pantaleoni, our use of RJMCMC allows us to more thoroughly reason about Jacobians, and enables us to design a perturbation with optimal (i.e. unconditional) acceptance rates.

6.6 Results

We validate our theory with two implementations of our proposed method. Our first implementation performs MCMC integration in a simplified 1D scenario. The purpose of this simulation is not to compare performance across different techniques, but to demonstrate the correctness of our approach. We find that the high dimensionality of the light transport problem and the use of large steps can often mask subtle biasing issues of Markov Chain rendering methods, whereas these are immediately apparent in the 1D case.

Our second implementation adds reversible jumps to a general ray tracing renderer as an auxiliary perturbation on top of Multiplexed MLT. We evaluate the performance of this implementation in detail in Section 6.6.2.
6.6.1 1D Markov Chain Integrator

Our simulation performs Markov Chain integration on the [0, 1] interval with a sinusoid target distribution (Fig. 6.3 (a), top). Each integration method has three sampling techniques at its disposal with PDFs corresponding to: a triangular distribution (a), a step function distribution (b), and a mixture distribution (c) of a sinusoid and a linear function, as illustrated below.

The primary sample space in this scenario has three dimensions: One used for selecting the currently used technique, one for sampling a position and one for selecting a subtechnique in mixture distributions (only utilized in the third sampling technique). We demonstrate the results for this setup in Fig. 6.3.

The histogram of the Markov Chain states (top row) should converge to the sinusoid target distribution if the algorithm is correct. We also record a secondary histogram that keeps track of which sampling techniques were used by the Markov Chain on average at particular points in space (bottom row). For the integration methods considered here, this histogram should converge to the MIS weights of the sampling techniques.

We implemented the equivalent of Multiplexed MLT in this 1D framework as a baseline integrator (Fig. 6.3 (a)), which represents the ground truth for both histograms. In order to amplify potential issues in the perturbations, we disable large steps for all integrators, and only allow the Markov Chain to explore the 1D space using small steps and reversible jumps.
Figure 6.3: In a simplified 1D scenario, we compute the stationary distribution (top row) and average use of three available sampling techniques (bottom row) for four different Markov Chain integrators. We demonstrate the correctness of our full approach (d) compared to the Multiplexed MLT ground truth (a). Neglecting the necessary Jacobians (b) or improperly parametrizing inverses (c) leads to skewed distributions, demonstrating the importance of our full theory.

To demonstrate the importance of correctly performing reversible jumps, we implement two versions of our algorithm that use only parts of our theory. The first integrator performs reversible jumps without taking into the account the Jacobian when transitioning between subspaces, and omits the right-most determinant term in Eq. (6.13). Unlike traditional perturbations, reversible jumps are fully deterministic aside from selecting one of the available inverses, and one may be tempted to treat them as a discrete transition with trivial acceptance probability. However, neglecting to incorporate the Jacobian of the transition leads to a heavily distorted stationary distribution (Fig. 6.3 (b)) and a skewed use of the different sampling techniques.

The second integrator includes Jacobians in the acceptance probability (Eq. (6.13)) and inverts sampling mixtures nearly correctly (Eq. (6.17)), but does not fully parametrize $\bar{M}_x$. Because the third dimension of primary sample space is used to select a subtechnique, an ambiguous interval arises when inverting the mixture distribution. Rather than parametrizing these intervals as described in Section 6.4.1, this integrator always returns a fixed point inside the interval. Although this approach
may not immediately appear incorrect, computing inverses in this manner leads to a biased distribution (Fig. 6.3 (c)).

Finally, we implemented our full approach, using both Jacobians and parametrized inverses. Incorporating our full theory leads to a correct stationary distribution, as shown in Fig. 6.3 (d).

### 6.6.2 General Ray Tracing Renderer

Our most general implementation adds RJMLT to an existing rendering system [9]. We implement reversible jumps as an additional perturbation on top of Multiplexed MLT. At each step, the Markov Chain selects either a large step, small step or a technique change with a fixed probability distribution (10%, 85% and 5%, respectively). Unlike Multiplexed MLT, we explicitly separate the technique index from the rest of primary sample space, such that small step perturbations may not change the sampling technique.

We tested our implementation on an array of 8 different test scenes, and gather a series of metrics to evaluate our approach. The scenes feature a wide variety of sampling methods, including layered materials, microfacet models, environment- and area lighting. Because MLT-based rendering methods can only compute a correct solution up to a global scaling factor, we run a path tracer on each scene for several hours to compute noise-free estimates of these factors and supply them to both MMLT and RJMLT. This allows us to exclude differences in scaling factors from the comparison.

The use of correlated samples can make a fair comparison between MCMC-based rendering methods challenging. Because Markov Chains generate most samples with local exploration, integration error can manifest in different forms, such as “splotches”, streaks, wrongly positioned caustics or entirely missing transport modes. The presence of many such artifacts is symptomatic of the Markov Chain “getting
stuck” in local maxima, and suggests that inappropriate perturbations are employed by the integrator. These artifacts have implications for both the visual appearance and quantitative convergence, and we provide a series of comparisons to highlight the differences between RJMLT and MMLT:

Convergence Plots. Inappropriate perturbations lead to a slower convergence of the Markov Chain to its target distribution. To perform a quantitative analysis of this fact, we track the MSE (compared to a path traced reference image) of both RJMLT and MMLT as a function of render time. Because of the use of correlated samples, a single run of an MCMC integrator may not be representative, and we run 50 instances of each integrator with different random seeds to get a measure of its average behavior. In addition, the variation of the MSE between runs is an indication of the stability of the integrator. We plot graphs of both of these statistics in Fig. 6.4. The thin line represents the average MSE over time, whereas the shaded area visualizes the standard deviation of the MSE across all 50 runs. In all test scenes, RJMLT has a lower average MSE than MMLT at equal render time, sometimes significantly. Additionally, RJMLT exhibits less variation between runs than MMLT, which suggests that our proposed perturbation is more effective at exploring potential light paths than MMLT small steps.

Acceptance Probabilities. A direct measure of the performance of a perturbation is the rate at which its proposals are accepted. We track the average proposal acceptance rates in the Living Room scene, and compare the results in Fig. 6.5, broken down over path length. We differentiate between three different perturbations: MMLT small steps that leave the sampling technique unchanged, MMLT small steps that change the technique, and our proposed RJMLT perturbation. RJMLT small steps and MMLT small steps without technique change have identical behavior, and we only show the MMLT results. As the path length increases, the acceptance rates of
small step proposals decreases rapidly. Additionally, MMLT proposals that change the technique index have a significantly lower acceptance probability, dropping as low as 1% at path length 10. In contrast, our proposed perturbations achieve average acceptance probabilities close to 100% across all path lengths. The slight drop for longer paths stems from perturbations that could not perfectly invert a path due to floating point inaccuracy, and were rejected by our algorithm.
Figure 6.5: We visualize the average acceptance probability of perturbations in the Living Room scene, broken down over path length. To demonstrate the benefit of our method, we differentiate between proposals that change the sampling technique, and proposals that leave it unchanged. For Multiplexed MLT, proposals that attempt to change the sampling technique have a significantly lower acceptance probability compared to proposals that do not, and it is difficult for the Markov Chain to transition between sampling techniques. Conversely, technique changes in Reversible Jump MLT are nearly always accepted.

**Equal-time renderings.** Finally, we also provide renderings of RJMLT and MMLT after 5 minutes of render time in Fig. 6.6. We show a subset of our scenes, and provide two insets for each scene highlighting interesting features. Because of the structured nature of MLT noise, it is difficult to judge renderings of the methods by their noise level alone. We have added arrows to the MMLT insets pointing out areas of significant error. These include e.g. streaks on the pots and cutting board in the Kitchen scene; streaky reflections and hard-edge shadows in Living Room; wrongly positioned or missing caustics in Bathroom; and significant splotches in Glass of Water.

### 6.7 Discussion

We conclude this chapter with a short summary of our contributions, and follow with a review of the limitations of our proposed algorithm.
Figure 6.6: We compare Multiplexed MLT (left) to Reversible Jump MLT (right) across 5 different scenes at equal render time. The scenes feature complex illumination and occlusion, glossy caustics and long light path lengths. Because of its ability to transition between sampling techniques easily, RJMLT has fewer artifacts (streaks, splotches, wrongly positioned caustics) than Multiplexed MLT at equal render time. Such artifacts are symptomatic of the Markov Chain “getting stuck” and exploring a small part of path space for too long.
Figure 6.7: Continuation of Fig. 6.6
6.7.1 Contributions

In this chapter, we presented the following contributions:

- We introduced the Reversible Jump MCMC framework to the field of light transport. This framework allows for a wide range of novel perturbations in primary sample space that transition between different subspaces, even of different dimensionality.

- We introduced the notion of inverse sampling techniques, which allows seamless transitions between path space and primary sample space. Realizing these inverses involves a number of theoretical and practical concerns, and we solve these challenges through a novel extended path space and distill it to practical and easily realizable recipes.

- We constructed a new perturbation, which we call Reversible Jump MLT, that allows the Markov Chain in Multiplexed MMLT to easily transition between sampling techniques. We perform an in-depth mathematical analysis of this perturbation, and derive the correct acceptance probability using the RJMCMC framework. Two implementations confirm the correctness of our method, and detailed metrics show improved temporal coherence, faster convergence and decrease in artifacts of our method compared to MMLT.

Although we focused on one particular application of RJMCMC and inverse sampling techniques, the existence of these techniques blurs the boundary between path space and primary sample space. Any MCMC rendering algorithm that natively lives in one space can now make use of perturbations that live in the other, by temporarily transitioning to the space and transitioning back after performing the perturbation. We believe this unlocks a number of exciting new rendering algorithms.
6.7.2 Limitations

While the theory discussed in this paper is general, there are a few types of sampling schemes that make path inversions impractical in certain cases. Our current implementation will simply reject perturbations that involve one of these sampling schemes. This preserves correctness, but the efficiency of the algorithm will suffer (i.e. revert to that of MMLT) if they make up the majority of the transport in scene.

Rejection sampling (Section 3.1.4.3) is an alternative recipe for deriving sampling algorithms for general target distributions. Sampling methods derived in this manner, such as Woodcock tracking [141], consume a potentially unbounded set of random numbers and are virtually impossible to invert. However, the gains obtained from applying PSSMLT based methods on top of such a sampling scheme are questionable, and we believe this not to be a practical issue.

Certain specialized sampling schemes (e.g. the Box-Muller transform [19]) produce more than one sample for one set of inputs. Surplus samples are usually cached and reused in future sample queries, introducing a dependence between paths. Given only a single sample, it is not clear how to invert such a scheme and how the correct acceptance ratio should be computed. The reverse is also possible: Sampling schemes such as RIS [128] (Section 3.1.4.2) produce multiple candidate samples and discard all but one to produce the output sample. Given the output sample, it is difficult to reconstruct the candidates.

Finally, implementing path inverses requires additional engineering effort for every sampling method used in the underlying rendering algorithm. Although individual inverses are easily derived, the number of sampling techniques present in practical rendering systems can pose a challenge. This is an inevitability of moving beyond an exclusive primary sample space view.
SELECTIVELY METROPOLISED MONTE CARLO
LIGHT TRANSPORT SIMULATION

Up to this point in this dissertation, we have introduced a number of new rendering algorithms (Chapter 5, Chapter 6). The goal of these algorithms, much like most related work in the field, is to reduce variance compared to their predecessors in challenging light transport scenarios. These techniques add to the long list of rendering algorithms available to practitioners today: Where only the simplest of light transport algorithms—path tracing—was available at the inception of the rendering equation \([78]\), there now exists a dizzying array of highly sophisticated rendering techniques for nearly any type of transport encountered today.

However, of these methods, it is path tracing that has found widespread adoption in industry \([25, 42]\), while more sophisticated techniques remain much less popular. There are likely several reasons for this. Sophisticated Monte Carlo methods target challenging transport scenarios that would be impractical to render with path tracing, but generally do not improve handling of “simple” transport, i.e. that already handled well by path tracing. Moreover, the additional complexity of modern transport methods comes at the cost of considerable computational overhead, and/or visual artifacts such as correlated noise. If the scene does not contain or is not dominated by difficult transport, then path tracing can significantly outperform more sophisticated methods when compared at equal render time.

Compounding this problem is that rendering algorithms can only be employed globally on all transport in the scene, and the decision of which algorithm to use
is left to the end user. It is difficult to predict \textit{a priori} the types of transport in the scene, and whether path tracing suffices or a more sophisticated rendering algorithm is warranted. For this reason, using path tracing exclusively is simply a pragmatic choice; its simplicity makes it easy to implement, and it can handle the majority of salient transport well. The remaining problematic transport is either avoided entirely through careful scene design, filtered out \cite{149} in a post-process or removed entirely \cite{36,148} during rendering.

In this chapter, we introduce a new algorithm to address this problem. Our idea is to \textit{selectively} combine a “simple” base algorithm, such as path tracing, with a more sophisticated algorithm, so that the base algorithm is used for all transport that it handles well, and the more sophisticated algorithm is only used where the base algorithm fails. The combined algorithm should inherit both the high performance of the base algorithm, as well as the robustness of the sophisticated algorithm; existing scenes already handled well by the base estimator should be rendered with nearly identical performance, while unlocking new scenes with challenging transport.

We achieve this by partitioning the integration domain into the “simple” and “difficult” transport, i.e. those handled well by the base estimator and those that are not. Building on our experience from Chapter 6, we use a Metropolis sampler to guide the sophisticated rendering algorithm to only estimate where it outperforms the base estimator. This partition can be achieved with Multiple Importance Sampling \cite{134}, and we show that the standard MIS heuristics can be interpreted as a crude firefly detector Section 7.1. The combination can be improved through the use of a more sophisticated firefly detector, and we discuss the design decisions needed to make the resulting MCMC estimator (Section 7.2) and outlier detector (Section 7.3) practical. Finally, we empirically compare our combined estimator to its components in Section 7.4.
7.1 Selective Metropolisation via MIS

The goal of our algorithm fundamentally is to optimally combine multiple Monte Carlo estimators. Earlier in this dissertation (Section 3.1.5), we already reviewed how multiple importance sampling is one possible solution for this problem. Assuming we generated $n_1$ samples with PDF $p_1$ from the sophisticated algorithm, and $n_2$ samples with $p_2$ from the base algorithm, then the balance heuristic weights for samples from the sophisticated algorithm would be

$$w_1(\tilde{u}) = \frac{n_1 \cdot p_1(\tilde{u})}{n_1 \cdot p_1(\tilde{u}) + n_2 \cdot p_2(\tilde{u})}, \quad (7.1)$$

and analogously for the base algorithm. Similarly, the maximum heuristic would assign the binary weights

$$w_1(\tilde{u}) = \begin{cases} 
1 & \text{if } n_1 \cdot p_1(\tilde{u}) > n_2 \cdot p_2(\tilde{u}) , \\
0 & \text{otherwise} .
\end{cases} \quad (7.2)$$

In theory, MIS already provides a mechanism for combining estimators with different strengths. However, MIS only provides a way for weighting samples from different estimators after they have already been generated; it gives us no way of directing either estimator to only generate samples in the domains in which they outperform the other.

In the previous chapter, we already saw how MIS weights can be incorporated into the target function of a Markov chain. This will allow us to focus the sophisticated algorithm to where it provides the most benefit. Methods based on MCMC, however, suffer from non-uniform noise and temporal artifacts that we do not want to inherit, so we need to first identify the scenarios in which MCMC performs well.
Monte Carlo methods (middle row) distribute their samples uniformly at random in primary sample space. This works well if the energy landscape is close to uniform (left column), but fails to sample small energy islands well (middle column). MLT (bottom row) performs poorly in flat regions (left column), but excels at exploring difficult paths (middle column). When a scene contains a mixture of both scenarios, neither strategy works well in isolation (right column).

In Fig. 7.1, we illustrate a simplified direct lighting scenario with varying occlusion. A naive Monte Carlo approach (middle row) in this scene performs well if large areas of uniform brightness of the emitter are visible (left), but does poorly if light is visible only through a small window (middle). In contrast, an MCMC based method performs well if the target distribution is non-zero only in a narrow region (middle), but does poorly on large areas with uniform importance (left). Neither method performs well on a scene with a mixture of both cases (right).

The goal of this section is to come up with an MIS weighting strategy $w(\mathbf{u})$ that assigns small weights to large, flat areas easily integrable with Monte Carlo and large weights to small islands that are best explored with MCMC. We will then use the
base estimator to integrate \( f(\bar{u})(1 - w(\bar{u})) \) in a straightforward way, and use MCMC to explore the target distribution \( f(\bar{u})w(\bar{u}) \), which makes it focus its sampling effort where MCMC performs well compared to the base estimator.

We begin by making the MIS weights in Eqs. (7.1) and (7.2) concrete for a combination of a Monte Carlo estimator with MCMC. For simplicity, we consider combining a Monte Carlo estimator with a “Metropolized” version of the same estimator using PSSMLT; however, the same weights hold for any MCMC method in primary sample space.

Both the max- and balance heuristics require knowing the PDF of the two estimators. In primary sample space, the PDF of the base estimator is trivially \( p_2(\bar{u}) = 1 \). The PDF of the Markov chain can be assumed to be the (normalized) target distribution, i.e. \( p_1(\bar{u}) = f(\bar{u})/c \), and the MIS weights for samples generated by the Markov chain become

\[
w_1(\bar{u}) = \frac{n_1 \cdot f(\bar{u})/c}{n_1 \cdot (f(\bar{u})/c) + n_2 \cdot 1} = \frac{f(\bar{u})}{f(\bar{u}) + n_2/n_1 \cdot c} = \frac{f(\bar{u})}{f(\bar{u}) + b'}
\]

(7.3)

for the balance heuristic, where \( b \) is some constant that does not depend on the sample, and

\[
w_1(\bar{u}) = \begin{cases} 1 & \text{if } n_1 \cdot f(\bar{u})/c > n_2 \cdot 1 \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } f(\bar{u}) > b' \\ 0 & \text{else} \end{cases}
\]

(7.4)

for the maximum heuristic and some other constant \( b' \).

Equations (7.3) and (7.4) allow us to perform MIS between any Monte Carlo estimator and its PSSMLT variant. The Markov chain will automatically move towards regions where the base MC estimator performs poorly, and high-variance samples generated by the MC estimator will be downweighted before being added to the estimate.
This forms the basis of a workable rendering algorithm. However, these specific weights are not without issues: For both heuristics, the MIS weight of the Markov chain depends solely on the brightness of the sample. In practice, this leads to counter-intuitive results, such as MCMC being used to explore bright direct lighting, and challenging, but dim, indirect lighting being left to the base estimator.

Although not ideal, the weight derived from the maximum heuristic provides a starting point for deriving a better weighting scheme: Looking at Eq. (7.4), this heuristic assigns zero weight to the Monte Carlo estimator whenever the sample weight exceeds some threshold, i.e. \( f(\bar{u}) > b \). If we did not use MCMC in addition, this scheme would be equivalent to discarding samples with high contribution ("fireflies") to avoid high variance at the cost of bias.

Classifying samples as fireflies based on brightness alone is the simplest in a family of outlier detectors. These algorithms detect unlikely Monte Carlo samples, and are usually employed as a last-resort measure to remove troublesome samples. However, outlier detectors form a strong candidate for a good weighting scheme for combining MCMC and Monte Carlo: Samples that are difficult to sample for the base estimator are generated rarely by definition, and directly correspond to outlier samples. On the other hand, “simple” transport in the eyes of the base estimator will be sampled often, and would exclusively generate inliers.

Following these insights, we propose the following algorithm: Given an outlier detector that assigns large weights \( w(\bar{u}) \) to unlikely samples of the base estimator, we integrate the function \( f(\bar{u}) \cdot (1 - w(\bar{u})) \) with the base estimator, and use MCMC to integrate the function \( f(\bar{u}) \cdot w(\bar{u}) \). We give a high-level overview of our method in Algorithm 2. The Monte Carlo aspect of this algorithm is straightforward, but there is a large design space for the Markov chain. In the next section, we will explore the specific details needed to turn this MCMC method into a practical and efficient rendering algorithm for our purposes.
Algorithm 2: High-level outline of our rendering algorithm

<table>
<thead>
<tr>
<th>Function renderOneSPP()</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 // Run Monte Carlo sampler</td>
</tr>
<tr>
<td>2 outliers ← {}</td>
</tr>
<tr>
<td>3 for i ← 0 to numPixels do</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

7.2 Practical MCMC on Sparse Distributions

Naively running an existing MCMC method on the target distribution \( f(\bar{u}) \cdot w(\bar{u}) \) works poorly in practice. By design, this target distribution is sparse and contains a large number of small, disconnected “islands”. Both globally exploring this state space and finding suitable initial states \( \bar{u}^{(0)} \) with non-zero contribution become extremely challenging. Traditional MCMC methods pick the initial state from a small pool of Monte Carlo samples [135], but by design it is unlikely any of them are outliers and have non-zero contribution.

Our solution to this problem is two-fold: First, we directly use discarded samples of the Monte Carlo estimator as initial states for the Markov chain. Samples with low MIS weights for the base estimator will receive large weights in the corresponding MCMC target distribution and therefore form suitable initial states for
the Markov chain. This ensures a large and continually increasing sample pool for MCMC to choose from. Second, because it is difficult for MCMC to globally explore the state space through mutations, we don’t expend any effort into trying to do so, and only use perturbations in the Markov chain. To still ensure full coverage of the state space, we use many short-lived chains (100-1000 perturbations) instead of a single long-lived chain, which greatly improves stratification and noise characteristics of MCMC [26].

Each complete transport path formed by the Monte Carlo estimator is tested by the outlier detector and recorded in an outlier pool if \( w(\bar{u}) \) is non-zero. For each chain, we randomly select one of the samples in the outlier pool as the initial state \( \bar{u}^{(0)} \) and run the Markov chain for \( k \) steps. Assuming an MCMC budget of \( n_1 \) samples, we may run \( \lfloor n_1/k \rfloor \) chains in total. At each step \( \bar{u}^{(i)} \) of the Markov chain, we record a value of

\[
\frac{1}{n_1} \cdot \frac{f(\bar{u}^{(0)})}{\text{pmf}(\bar{u}^{(0)})} \cdot \frac{f_{\text{RGB}}(\bar{u}^{(i)})}{f(\bar{u}^{(i)})},
\]

in the framebuffer, where \( f(\bar{u}^{(0)}) \) is the brightness of the initial outlier, and \( \text{pmf}(\bar{u}^{(0)}) \) is the probability of picking that outlier as the initial state from the set of recorded outliers. To ensure all chains have approximately equal contribution, we set \( \text{pmf}(\bar{u}^{(0)}) \propto f(\bar{u}^{(0)}) \).

Similar to Cline et al. [26], we did not find our setting of the chain length parameter to be critical. Shorter chains generally lead to increased correlation artifacts, while longer chains lead to increased noise as fewer seed paths can be explored. We use 1000 mutations as a sane default, and did not need to adjust it for different scenes.
7.2.1 Allocating Samples

To make our algorithm effective, we need to appropriately budget the number of samples that the MCMC- and the Monte Carlo estimator should take. Too few samples for MCMC means we do not improve much over the Monte Carlo estimator alone, and too many increases render times without much decrease in variance.

The weighting introduced by the outlier detector effectively creates a soft partitioning of the integrand into two domains, where \( w(\bar{u}) \) is 1 or 0 respectively. The ideal sampling PDF in Monte Carlo is perfectly proportional to the integrand; therefore, it would on average place a number of samples in each domain proportional to the integral over the domain. That is, if the total number of samples taken is \( n \), then the region handled by MCMC should receive

\[
  n_1 = n \cdot r \quad \text{with} \quad r = \frac{\int_{\bar{u}} w(\bar{u}) f(\bar{u}) \, d\bar{u}}{\int_{\bar{u}} f(\bar{u}) \, d\bar{u}}
\]  

(7.6)

samples to match the distribution of the ideal PDF.

The actual number of outlier samples generated by the base estimator will differ from the ideal fraction \( r \) (that is what makes them outliers in the first place), and we use the MCMC estimator to compensate for this imbalance by adding additional samples exclusively in the outlier region. Knowing that \( n = n_1 + n_2 \), the number of MCMC samples is then determined by \( n_1 = n_2 \cdot r / (1 - r) \).

The number \( n_2 \) of MC samples is determined by the user (via samples-per-pixel), but we do not know the ideal sample ratio \( r \) \textit{a priori}. However, we note that the unweighted MC samples are an estimator of the denominator of Equation 7.6, and the weighted samples an estimator of the numerator. We therefore keep running averages of these two quantities and use their ratio to estimate \( r \) incrementally during rendering.
7.3 Detecting Outliers

In theory, our algorithm is not limited to any particular outlier detector. However, the choice of detector has a large influence on the practical efficiency of the algorithm, and if our goal is to obtain a fast renderer, this imposes several design constraints on the choice of outlier detector.

The primary goal of our algorithm is to not degrade the performance of the base integrator when it works well. In particular if path tracing is used as the base, the cost of obtaining a sample is comparably low; since the outlier classifier is run on each sample, the computational efficiency of the classifier is paramount. In addition, high-quality renderings require a large number of samples, and therefore the memory footprint of the classifier should be independent of the sample count. These constraints make outlier detectors operating in image space an attractive choice.

Pixels may contain samples from many independent integrals (multiple path lengths, light sources, etc.). Sample statistics for a pixel thus usually contain multiple modes, and the outlier detector should be able to represent these; we found simple approaches such as per-pixel mean and variance to not be sufficient.

This initially led us to consider Density-Based Outlier Detection (DBOR) \[36\]. While DBOR is an acceptable choice for the purposes of our algorithm, it does not have bounded memory footprint and its computational cost causes considerable efficiency loss on simple scenes (such as e.g. a Cornell box). More recently, Zirr et al. \[148\] introduced an outlier detector based on cascaded framebuffers. While similar in spirit to DBOR, this detector is computationally inexpensive, has bounded memory footprint, and is very effective at outlier detection, all the while being a very simple algorithm to implement. From a practical standpoint, this makes it an attractive choice for our algorithm.
Figure 7.2: We illustrate our chosen outlier detector in a scene containing difficult lighting. Starting from the raw Monte Carlo samples (top row), we first record individual samples in one of several cascaded frame buffers based on sample brightness (middle row). Using the number of samples with similar brightness as an indicator of whether a sample is unlikely, the detector reliably distinguishes paths handled well by Monte Carlo (bottom left) from outlier paths (bottom right) on a per-sample basis.
7.3.1 Implementation Details

In the following, we first describe the detector of Zirr et al. [148] (which we illustrate in Figure 7.2) in detail, before motivating our particular choice of parameters.

This detector stores a set of exponentially spaced framebuffers \( B_0, \ldots, B_m \), in which buffer \( B_j \) at each pixel counts the number of samples that fall in a luminance range centered on \( \alpha \cdot \beta^j \). All samples generated by the Monte Carlo estimator, outlier or not, are recorded in the cascaded framebuffer. For a sample with luminance \( x \), we find the nearest buffers \( B_j \) and \( B_{j+1} \) such that \( j = \lfloor \log_{\beta} x/\alpha \rfloor \), and add a value of \( w = \log_{\beta} (x/\alpha) - j \) to the corresponding pixel in \( B_j \) and a value of \( 1 - w \) to \( B_{j+1} \). To determine if a sample is an outlier, we linearly interpolate from the nearest buffers using the same weights as with splatting, and check if the interpolated value is smaller than a threshold \( \tau \).

Choice of Parameters. High-energy samples are almost exclusively outliers, and we consider samples far beyond meaningful image values (1000 in our implementation) automatically as outliers. The detector only needs to operate in the regime below this threshold, and we pick the number \( m \) of buffers to cover the luminance range below 1000.

The exponential spacing of the buffers controls the discretization of the luminance range: A smaller spacing means a larger number of modes can be represented per pixel; but it also means per-buffer statistics are noisier at lower sample counts. We choose \( (\alpha = 1/2, \beta = 2) \) as a tradeoff between robust detection at low sample counts and luminance resolution. This differs from the powers of 8 used by Zirr et al. [148]; this is because they focus on extremely high-energy fireflies and require all samples to be splatted to a buffer, thus the larger dynamic range. Our method benefits from discriminating lower energy outliers and gracefully handles samples brighter than the maximum representable luminance, motivating the use of a smaller spacing.
We set the classification threshold $\tau = \max(3, \text{spp/m})$, which considers buffers that receive less than uniform ($< \text{spp/m}$) samples to likely contain outliers. At low sample counts we do not have much information; we err on the side of caution and clamp the threshold to a lower bound.

**Updating the Detector.** Updating the outlier detector while simultaneously using it to classify outliers would lead to bias, as MIS weights would shift during rendering. We instead render the image in iterations of a small number of samples per pixel at a time. We keep two copies of the outlier detector: The first is used only for classifying outliers and is invariant during each iteration, while the second records new samples generated from the Monte Carlo estimator. After the iteration is complete, we copy the contents of the second detector into the first.

**Additional Thresholding.** Even if a sample is classified as an outlier, it may not be worth exploring with MCMC. This is because the probability of an outlier being chosen as a chain seed depends on its sample score; very dim outliers are both picked rarely, and do not contribute much variance, that it is better to record them in the image directly than to hope for them to become a Markov chain. To account for this, we filter outliers before inserting them into the pool of MCMC seeds with simple thresholding of their sample score. We determine the threshold automatically during rendering: Between each iteration, we identify the outliers whose probability of being chosen for MCMC exploration was smaller than some threshold ($1/2$ in our implementation). We consider these samples not worth exploring, and set the threshold for the next iteration to be the maximum sample brightness within this set.
7.4 Results

We implemented our method in the open source rendering system Tungsten [9], and tested it on a diverse test set of indoor- and outdoor scenes that contain varying mixtures of difficult and simple light transport. Although our method could be used with any Monte Carlo and MCMC estimator, we found the combination of path tracing and RJMLT to work the best, and only show results using this combination in the paper. We rendered time-sensitive results on an 8-core AMD Ryzen 1800x with 16 threads, and all remaining results on a heterogeneous compute cluster. We evaluate our method using a variety of metrics, detailed in the following paragraphs.

Equal-time Renderings. In Fig. 7.3, we show several scenes rendered with path tracing, RJMLT, our method and a modified version of ERPT that uses RJMLT. Sample counts were normalized so all methods complete in equal time of 150 seconds. We also show the MSE of each method to a reference, relative to our method. This corresponds to how much longer we should expect to render with each method to obtain the same error as our method. In nearly all scenes, our method provides improvements of 2–15× over the estimators our method is combining, both visually and in terms of MSE. The only exception is the pool scene: Because the image is dominated by very difficult paths, the computational effort expended on path tracing does not provide much benefit over running RJMLT alone, and our method performs slightly (0.8×) worse than RJMLT alone.

Convergence Plots. For MCMC based methods, individual renderings may not be representative of the behavior of the algorithm due to correlations. For each estimator, we therefore also rendered 30 independent runs with different random seeds, and computed the evolution of MSE over time. The average MSE then provides
Figure 7.3: We show a diverse set of scenes rendered with path tracing, ERPT with reversible jumps, RJMLT and our method, and compare them at equal render time. We show the relative MSE (computed over the entire image) of each method compared to ours, which is equivalent to how much longer each method would have to render to reach the same MSE as ours. Our method shows significant noise reduction compared to prior work.
Figure 7.4: We measure the MSE of our method and previous work over 30 independent runs and visualize both the mean MSE (thick curve) as well as the standard deviation of the MSE (shaded regions) over equal time. A large shaded region means that the MSE fluctuates significantly between runs, indicative of severe temporal flickering. On most scenes, our method has both significantly lower MSE as well as less variation across runs compared to previous work. Note that because of logarithmic axes, the shaded region is not symmetric around the mean.
a more stable comparison metric, which we plot in Fig. 7.4. We also compute the standard deviation of the MSE across all runs, which gives an indication of the temporal flickering for each method. We visualize this with a shaded region around the average MSE corresponding to one standard deviation. Note that because of logarithmic axes, this region is asymmetric around the mean. As expected from a pure MCMC method, RJMLT has erratic convergence and significant variance between runs, which makes it undesirable for practical use. ERPT+RJMLT has smoother convergence and less temporal variance than pure RJMLT, but converges significantly slower. Much like path tracing, our method provides smooth convergence and temporal stability, but with significantly lower average error. In most scenes, our method significantly outperforms RJMLT alone, except when the scene is dominated by complex transport. At the same time, we offer significantly more stable convergence, and outperform path tracing by a wide margin.

7.5 Discussion

We conclude this chapter with a short summary of our contributions, and follow with a review of the limitations of our proposed algorithm.

7.5.1 Contributions

In this chapter, we presented the following contributions:

- We introduced a new rendering algorithm that can selectively combine a simple, but efficient Monte Carlo estimator with a more sophisticated, but robust rendering algorithm. We achieved this by guiding the second estimator with a Metropolis sampler towards regions where it outperforms the base estimator, measured by a specially crafted MIS weight.
We show that in primary sample space, the balance- and maximum heuristic in this case reduce to functions solely based on the sample brightness in primary sample space, which turns the maximum heuristic into a crude firefly detector.

We alleviate the problems of the standard weights by using a more reliable outlier detector, and discuss the design decisions needed to obtain a practical, parameter-free rendering algorithm that is robust and efficient.

Our final method significantly outperforms both path tracing and RJMLT on the majority of our test scenes, with significantly reduced temporal flickering and improved noise characteristics compared to standard MCMC methods.

7.5.2 Limitations

There are some limitations to our work. Firstly, we rely on a simple MC estimator to produce samples for the Markov chain to explore. If the transport is too difficult for the MC estimator to sample, then it will not produce enough seed samples for the MCMC portion of our method to work effectively. This problem can be circumvented by using a better MC base method (like bidirectional path tracing), but this defeats our goal of avoiding experimentation with different integrators or parameters for each scene.

The Dragon scene represents a failure case of our method. Here, our method spends computational effort exploring glossy-glossy interreflections, which slightly reduces noise but disproportionately increases render times, and our method performs approximately $0.5 \times$ as well as path tracing. We did no parameter tuning or much optimization on our method, and it is possible that additional work may close this gap. Better perturbation strategies or better criteria for which samples to explore might also improve the performance of our method on the simplest of scenes. However, our method still performs reliably on scenes with more realistic complexity.
Although our outlier-based weighting scheme works well in practice, it currently lacks a more principled analysis. The variance proofs of the balance- and maximum heuristic only hold under sample independence, and it would be instructive to derive new heuristics that explicitly incorporate not just the PDFs, but also the sample correlations and computational cost to minimize variance. These weights may be able to more readily explain when MLT works better than standard Monte Carlo, and when it does not.
In chapters 5–7, we have introduced several new rendering algorithms that exploit reuse of computation to improve efficiency. In our extensions to photon points and beams, light subpaths are reused over many pixels for amortization; in our work on MCMC-based methods, high-contribution transport paths are reused in a Markov chain to incrementally sample related paths.

The methods we considered up to this point were squarely in the realm of offline rendering, where there are no hard constraints on the amount of time spent rendering an image. In this cost landscape, the evaluation metric for a rendering algorithm is primarily its asymptotic convergence in comparison to prior work, and the absolute cost of reusing a sample is not of much concern.

In this chapter, we will shift our focus to the problem space of real-time rendering. Contrary to before, there are now constraints on permissible render times to be within the real-time (<33ms) to interactive (≈ 100ms) regime, in addition to hard constraints on the resources—such as memory footprint, memory bandwidth, etc.—available to the rendering algorithm.

A metric of particular interest in this chapter is the number of ray-tracing operations required, on average, to render a complete image. In contrast to offline rendering, tracing rays in complex scene geometry has only recently become feasible.
in real-time thanks to the introduction of dedicated ray-tracing hardware. Even so, current-generation hardware only permits few (in the single-digit regime) rays to be traced per pixel at real-time rates, which severely restricts the set of rendering algorithms available.

In particular, the rendering algorithms we have introduced so far do not provide much benefit under these constraints. Common to these methods, and most related rendering algorithms in graphics, is that each reuse of a sample (e.g. each intersection with a photon plane, or each step in a Markov chain) leads to a new complete transport path, whose path contribution is evaluated from scratch. Evaluating a transport path involves tracing rays, either to perform a random walk (as in PSSMLT-based methods) or to evaluate visibility (as in nD photons). Because ray-tracing operations are a limited resource in real-time rendering, these algorithms would only allow very limited reuse of computation, and thus, little efficiency gain.

With these constraints in mind, we will introduce a new rendering algorithm in this chapter which performs reuse of samples across space (within an image) and time (across images in an animated sequence) to incrementally improve sample distributions at each pixel. By drawing only few samples from this distribution, only few paths need to be evaluated and thus few rays traced for each pixel; however, the distribution of these samples is optimized through large-scale reuse of computation, leading to low variance within real-time constraints.

To make the problem tractable, we focus our attention on estimating direct lighting only. While only a fraction of the complexity of full light transport, direct lighting remains a significant challenge in real-time rendering, and we will introduce an algorithm that is suited to real-time ray tracing with many emitters and fully dynamic scenes to address this challenge. We introduce the problem setting in Section 8.1 and discuss preliminaries in Section 8.2, before motivating and describing our approach at a high level in Section 8.3 and Section 8.4. We then perform a more rigorous derivation of the described estimator in Section 8.5, including detailed analysis of
the estimator’s bias. We give details of how our algorithm is implemented in practice (Section 8.6), before concluding with an analysis of its performance compared to prior work in Section 8.7.

8.1 Problem Setting

In this chapter, we primarily focus on the problem of estimating direct lighting. Given a sensing point \( y \) and outgoing direction \( \omega \), the directly reflected radiance \( L_r \) leaving \( y \) in direction \( \omega \) is

\[
L_r(y, \omega) = \int_A \rho(y, y\hat{x}, \omega) L_e(x, x\hat{y}) G(y, x) V(y, x) dA(x),
\]

which is a simplified version of Eq. (2.31) that considers only directly emitted radiance and assumes the absence of participating media. Of note is the presence of visibility function \( V \): Evaluating the integrand, such as in Monte Carlo estimation, thus requires tracing rays. To make this fact explicit, we have redefined \( G \) slightly and pulled out the visibility term.

We further restrict transport paths to be of length 2, such that \( y \) is connected to the sensor with a single segment. In this setup, all estimation points \( y \) are directly visible to the camera, and we can reason about neighborhoods of estimation points in screen space, i.e. in terms of pixel neighborhoods.

We will use a straightforward primary Monte Carlo estimator of Eq. (8.1),

\[
L_r(y, \omega) \approx \frac{f_q(x)}{p(x)} \quad \text{with} \quad f_q(x) = \rho(y, y\hat{x}, \omega) L_e(x, x\hat{y}) G(y, x) V(y, x),
\]

where \( y \) and \( \omega \) are uniquely determined by the pixel \( q \) in the rendering for which direct lighting is estimated, and \( x \) is a random variable over locations on emitting surfaces.
Although $f_q$ in this restricted scenario contains fewer terms than the general path contribution, finding suitable sampling PDFs $p$ that lead to low-variance estimates remains a major challenge.

8.2 Resampled Importance Sampling

Resampled Importance Sampling (RIS) [128], which we first reviewed in Section 3.1.4.2, forms the foundation of the rendering algorithm we introduce in this chapter. We will reintroduce it here in slightly rearranged form to simplify later analysis of the algorithm (Section 8.5).

Given a set of candidate samples $x = \{x_1, \ldots, x_M\}$ from a source distribution $p$, RIS can produce samples approximately distributed following a desired target distribution $\hat{p}_q$. Remarkably, RIS only requires knowing $\hat{p}_q$ up to a proportional factor, which permits approximately sampling distributions that are not easily integrable. The distribution of the samples returned by RIS converges to the target distribution as $M \to \infty$, where the speed of convergence is determined by how closely $p$ approximates $\hat{p}_q$.

Conceptually, RIS is not dissimilar from the Markov Chain Monte Carlo algorithms we considered in Chapter 6 and Chapter 7, which also produce samples distributed approximately proportional to an arbitrary target distribution. Different to these methods however, the PDF of the samples produced can be readily estimated without knowing the normalization factor of $\hat{p}_q$ (which MCMC methods require). This means RIS samples can be used in primary Monte Carlo estimators without modification.
Given the candidate samples and their PDF, as well as the desired target distribution, RIS selects an output sample \( y = x_z \) from the candidates with probability

\[
p(z \mid x) = \frac{w(x_z)}{\sum_{i=1}^{M} w(x_i)} \quad \text{with} \quad w(x) = \frac{\hat{p}_q(x)}{p(x)}.
\]  

(8.3)

The PDF of the sample \( y \) is not available directly, but for now we will assume that we have access to a function \( W(y) = 1/p(y) \) that returns the reciprocal of the PDF of the sample; we analyze this function in more detail in Section 8.5. Given \( W \), we can form the primary estimator

\[
Y = f_q(y) \cdot W(y).
\]  

(8.4)

**Discussion.** This estimator is ideal for our purposes for several reasons. In our scenario, we can only afford few evaluations of \( f_q \) due to the limited number of rays that can be traced at real-time framerates. However, the RIS estimator allows us a large amount of flexibility in the distribution of samples used in the primary estimator: Any distribution \( \hat{p}_q \), even if it cannot be sampled or integrated through conventional means, can be approximately sampled with RIS. As long as \( \hat{p}_q \) closely approximates \( f_q \), we can obtain very low variance primary estimators not attainable through conventional importance sampling methods. This means fewer evaluations of the primary estimator—and thus, fewer rays traced—to obtain comparable variance to prior work.

The choice of \( \hat{p}_q \) is critical to the efficiency of RIS. If \( \hat{p}_q \) is too dissimilar from \( f_q \), then we will not be able to obtain a low variance primary estimator, even in the limit of \( M \to \infty \). Simultaneously, if \( \hat{p}_q \) is too dissimilar from \( p \), a large value of \( M \) is required to converge to the desired distribution, and both storing and evaluating large numbers of candidate samples becomes impractical. We opt to sidestep this problem by choosing

\[
\hat{p}_q(x) = \rho(y, \bar{y} \vec{x}, \omega) L_e(x, \bar{y}) G(y, x)
\]  

to contain all terms of \( f_q \)
save for visibility (i.e. all terms that do not require tracing rays). To solve the problem
of slow convergence to the target distribution with respect to $M$, we introduce several
modifications to RIS over the next few sections that allow us to attain large effective
values of $M$ while maintaining real-time rates.

### 8.3 Reservoir Resampling

A first hurdle to applying RIS to real-time rendering is that it requires selecting a
sample from a set of candidate samples with probability proportional to some weight
$w(x)$. A straightforward implementation would generate all candidate samples up-
front, tabulate their cumulative mass function and numerically invert it \[110\].

The problem with this approach is that it requires temporarily storing all can-
didates. Our goal is to target dedicated graphics hardware, which runs many (> 10’000)
invocations of the rendering algorithm in parallel. Even for modest $M$, the cumu-
lative storage requirements of the candidates over all invocations can exceed the
capacity of fast memory on current hardware.

To solve this problem, we import the family of weighted reservoir sampling (WRS) \[24, 137\] algorithms to graphics. WRS can select an element $x_z$ from a stream
$x_1, x_2, \ldots, x_M$ of data in a single pass, using only a constant amount of storage.
Each element has associated with it a weight $w(x_i)$, such that $x_i$ is selected with
probability

$$p_i^M = \frac{w(x_i)}{\sum_{j=1}^{M} w(x_j)}. \quad (8.5)$$

The stream length $M$ does not have to be known in advance; the stream can be
terminated at any point, and the output element is distributed exactly with $P_i^M$ over
all input elements observed thus far.
Algorithm 3: Weighted reservoir sampling.

```java
class Reservoir
{
    y ← 0  // The output sample
    w_{sum} ← 0  // The sum of weights
    M ← 0  // The number of samples seen so far

    function update(x_i, w_i)
    {
        w_{sum} ← w_{sum} + w_i
        M ← M + 1
        if rand() < (w_i/w_{sum}) then
            y ← x_i
    }

    function reservoirSampling(S)
    {
        Reservoir r
        for i ← 1 to M do
            r.update(S[i], weight(S[i]))
        return r
    }
}
```

Many variations of reservoir sampling exist, depending on whether only one or many output elements are desired, and whether elements should be selected with or without replacement. For our purposes, the simplest of these algorithms—choosing a single element with replacement—is sufficient.

WRS maintains a reservoir that contains the currently selected element, along with internal state to allow correct selection. At each point in the input sequence, the element in the reservoir satisfies the invariant that it is drawn from the desired distribution over all elements processed thus far. When processing a new element, the element in the reservoir is replaced with the new element with some probability $P$, where $P$ is selected such that the reservoir maintains the invariant.

More precisely, after processing $m$ samples, the probability of sample $x_i$ ($i \leq m$) appearing in the reservoir is exactly $P^m_i$. After processing element $x_{m+1}$, this invariant must be maintained. Enforcing it for element $x_{m+1}$ immediately tells us that $x_{m+1}$ should replace the element in the reservoir with probability $P = P^{m+1}_{m+1}$. Any previous sample $x_i$ ($i \leq m$) now appears in the reservoir with probability

$$P^m_i \cdot (1 - P) = \frac{w(x_i)}{\sum_{j=1}^{m} w(x_j)} \left(1 - \frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)}\right) = \frac{w(x_i)}{\sum_{j=1}^{m+1} w(x_j)}, \quad \text{(8.6)}$$
Algorithm 4: Reservoir resampling.

```plaintext
foreach pixel q ∈ Image do
  Image[q] ← shadePixel(RIS(q), q)

function RIS(q)
  Reservoir r
  for i ← 1 to M do
    generate x_i ∼ p
    r.update(x_i, ˆp_q(x_i)/p(x_i))
  return r

function shadePixel(Reservoir r, q)
  return f_q(r.y) · r.W(r.y)
```

which also maintains the invariant. The correctness of the algorithm then follows by induction.

We show a pseudo-code implementation of this algorithm in Alg. 3. This algorithm was introduced by Chao [24], and only requires storage for the sample in the reservoir and a running sum of weights to compute \( P \). Of note is that, although WRS is relatively unknown in graphics, this algorithm has been independently reinvented several times [93, 143].

WRS is a near perfect solution to the problem we outlined at the beginning of this section. We combine RIS and WRS into a streaming algorithm we term reservoir resampling (Alg. 4) that completely removes the need to store the \( M \) candidates of RIS. Instead, candidate samples are generated individually, processed through the reservoir and immediately discarded.

**Motivating Example.** In Fig. 8.1, we show an image from a GPU implementation of reservoir resampling for direct lighting in a complex scene with 23,000 emissive triangles, where the distribution of candidates is uniform over the area of all emitters. We compare this algorithm with varying candidate counts \( M \) to a state-of-the-art real-time light sampling algorithm [103] at an equal number of rays traced per pixel. Surprisingly, as \( M \) increases, reservoir resampling—despite the relatively poor can-
8.4 Reservoir Reuse

In the previous section, we introduced reservoir resampling as a practical algorithm for performing RIS on current graphics hardware. The combination with WRS reduced the storage requirements from $O(M)$ to $O(1)$; however, the computation required is still $O(M)$. If the target distribution $\hat{p}$ is too dissimilar from the candidate distribution $p$, the number of candidates required for a low-variance estimate may be intractable at real-time rates.

Figure 8.1: The quality of reservoir resampling improves with increased $M$ (candidates) and $N$ (samples of secondary estimator). Here we show the effect of increasing $M$ in the multi-room subway scene with 23,000 textured emissive triangles. Tracing 8 shadow rays costs 6 ms; selecting those samples costs (left to right) 1.0, 2.5, 10.1, 42, and 168 ms. Moreau et al. [103]'s total cost is 48 ms when shooting 8 rays, comparable to $M = 1024$, but with quality comparable to $M = 256$. Moreau et al. [103]'s candidate distribution—outperforms even a state-of-the-art light sampling technique, without requiring any preprocessing or complex data structure.
In this section, we introduce a set of techniques based on reuse and sharing of samples between reservoirs to dramatically reduce the computational effort required. A key observation is that the distribution of samples produced by RIS converges to the target distribution faster (w.r.t. $M$) as the candidate distribution approaches the target distribution.

Sampling our chosen target distribution directly is not possible due to its complexity. However, we already know a general recipe for sampling any distribution approximately—it is RIS! We exploit this fact and propose to produce the candidate samples themselves by RIS to accelerate the convergence of RIS.

Similar in spirit to MCMC-based algorithms (Chapter 6, Chapter 7), we make this idea practical by recognizing that neighboring pixels in a rendering solve very related integration problems. If the integrands (and hence desired target distributions) of pixels in a neighborhood are not too dissimilar, then samples produced by RIS at any one pixel form excellent candidate samples for RIS at other pixels in the neighborhood.

This forms the basis of our algorithm of reservoir reuse. We begin by performing RIS at each pixel individually, using some predetermined number of candidates $M$. We then perform successive stages of RIS that use the samples from neighboring...
reservoirs in prior stages as candidates, where neighborhoods extend across space (within a rendering) or time (across renderings in an animated sequence). We show a high-level overview of our algorithm in Fig. 8.2.

In Alg. 5, we show how to perform RIS using samples from other reservoirs as candidates. Fundamentally, the algorithm is identical to RIS in Alg. 4, except that candidates are now reused from other reservoirs instead of generated from scratch, and the candidate PDF $p$ is replaced by the reciprocal PDF of RIS, $W$. All our proposed reuse operations make use of this algorithm, and only differ in which reservoirs are

![Algorithm 5: Combining the streams of multiple reservoirs.](image)

**Algorithm 5**: Combining the streams of multiple reservoirs.

**Input**: Reservoirs $r_i$ to combine.

**Output**: A combined reservoir using each of the reservoir’s output samples as candidates.

```plaintext
1 function combineReservoirs(q, r_1, r_2, ..., r_k)
2    Reservoir s
3    foreach r ∈ {r_1, ..., r_k} do
4        s.update(r.y, \( p_q(r.y) \cdot r.W(r.y) \cdot r.M \))
5        s.M ← r_1.M + r_2.M + ... + r_k.M
6    return s
```
combined. We track the effective number of candidates $M$ each reservoir received, and use it as a multiplicative weighting factor. This prefers reservoirs that are closer to their target distribution, and leads to significant variance reduction.

**Spatial reuse.** In Fig. 8.3, we show the benefit spatial reuse in the Subway scene. In each stage of spatial reuse, we select $k$ random pixels from a small spatial neighborhood around the current pixel, and use their reservoirs as candidates for an additional stage of RIS. Each iteration requires little additional computation, but dramatically increases image quality. The benefit is not indefinite; eventually, iterative reuse incorporates all candidates from nearby pixels and image quality stops improving.

**Temporal reuse.** In real-time applications, images are rarely rendered in isolation, but are part of an animated sequence. Scenes usually change smoothly over time, and the target distribution of a pixel in the previous frame is a good match to the
target distribution of the same pixel in the current frame. After rendering a frame, we store each pixel’s final reservoir for reuse in the next frame. If we render frames sequentially and feed forward their reservoirs, a frame combines candidates not just with those of the previous frame, but all previous frames in the sequence, which dramatically improves image quality. Figure 8.4 again shows the SUBWAY scene, comparing spatial-only and spatiotemporal reuse.

**Visibility reuse.** Unfortunately, even with an unlimited number of candidates, RIS cannot achieve noise-free renderings. Even as the distribution of samples approaches the target PDF \( \hat{p}_q \), \( \hat{p}_q \) does not sample the integrand \( f_q \) perfectly due to the missing visibility term. Because residual noise due to visibility can be severe in large scenes, we also introduce a stage of visibility reuse to solve this issue. Before executing spatial or temporal reuse, we perform an additional stage of RIS at each pixel that uses the full contribution \( f_q \), including visibility, as the target function; however, we only use the pixel’s own reservoir as the sole candidate. This means only a single ray has to be
traced (for that pixel reservoir’s sample $y$), and RIS essentially corresponds to conditionally discarding the sample if $y$ is occluded. This prevents occluded samples from propagating to neighboring pixels in the subsequent spatial and temporal stages. If visibility is locally coherent, then noise due to visibility is substantially reduced at the cost of a single additional ray per pixel. In Fig. 8.5, we show the same Subway scene, comparing spatiotemporal reuse with and without visibility reuse.

Alg. 6 provides pseudocode for our complete algorithm. We first generate and resample $M$ independent candidates at each pixel. The selected samples from this step are tested for visibility, and occluded samples discarded. We then combine the selected samples in each pixel’s reservoir with the prior frame’s output, determined using backprojection. We perform $n$ rounds of spatial reuse to leverage information from a pixel’s neighbors. Finally, we shade the image and forward the final reservoirs to the next frame.

8.5 Eliminating Bias

In the previous section we introduced, at a high level, a practical algorithm to reuse computation spatially and temporally that dramatically improves the quality of RIS with low overhead. However, we ignored one important detail: Much like all Monte Carlo estimators, RIS requires that the candidate distribution be non-zero wherever the target distribution is non-zero. However, this is potentially violated when we reuse candidates from neighboring pixels: Each pixel uses a different integration domain and target distribution, and neighboring pixels may not generate samples over the entire domain where $\hat{p}_q$ at the current pixel is non-zero.

In this section, we perform an in-depth analysis of the source of the bias, and how to eliminate it. We begin in Section 8.5.1 by first deriving the PDF of the RIS estimator explicitly. The resulting expression does not have a closed-form solution,
8.5 eliminating bias

Algorithm 6: Our algorithm for RIS with spatiotemporal reuse.

Input: Image sized buffer containing the previous frame’s reservoirs
Output: The current frame’s reservoirs

\begin{algorithm}
\begin{algorithmic}[1]
\Function{reservoirReuse}{prevFrameReservoirs}
\State reservoirs ← new Array[ImageSize]
\ForEach{pixel \( q \) ∈ Image}
\State reservoirs[\( q \)] ← RIS(\( q \)) \Comment{Alg. 4}
\EndFor
\EndFunction
\end{algorithmic}
\end{algorithm}

and we instead expose \( W \) to be an unbiased estimator of the reciprocal PDF of RIS in Section 8.5.2. This estimator becomes potentially biased when mixing multiple distributions, and we show two techniques for how it can be debiased in Section 8.5.4. Readers less interested in theory can skip directly to Section 8.5.5, in which we detail the practical changes to our algorithm needed to accomodate our theory.

8.5.1 The RIS PDF

We will now derive the effective PDF \( p(y) \) of samples produced by RIS. Standard RIS [128] (Section 3.1.4.2) assumes that all candidate samples are produced by the
same PDF \( p \). We instead now allow each sample \( x_i \) in \( x \) to come from a potentially different source PDF \( p_i(x_i) \). The joint PDF of these proposals is simply the product of their PDFs:

\[
p(x) = \prod_{i=1}^{M} p_i(x_i) . \tag{8.7}
\]

In the second stage of the RIS algorithm, we pick a discrete index \( z \in \{1, \ldots, M\} \), but with selection probabilities and weights now driven by these candidate-specific PDFs (cf. Eq. (8.3)):

\[
p(z \mid x) = \frac{w_z(x)}{\sum_{i=1}^{M} w_i(x_i)} \quad \text{where} \quad w_i(x) = \frac{\hat{p}(x)}{p_i(x)} . \tag{8.8}
\]

Since we have \( p(x) \) and \( p(z \mid x) \), we can easily write down the joint PDF of the candidates \( x \) and selected index \( z \) as the product:

\[
p(x, z) = p(x) \ p(z \mid x) = \prod_{i=1}^{M} p_i(x_i) \frac{w_z(x)}{\sum_{i=1}^{M} w_i(x_i)} . \tag{8.9}
\]

Given this joint PDF, we can now derive the marginal PDF \( p(y) \): For a fixed output sample \( y \), there are potentially many configurations of \( x \) and \( z \) that could lead to \( y \) being returned by RIS. For example, we could have \( x_1 = y \) and \( z = 1 \) and all other \( x_2, \ldots, x_M \) chosen freely. We could also have \( x_2 = y \) and \( z = 2 \), and so forth. Of note is that \( y \) can only be produced by techniques for which \( p_i(y) > 0 \). We will gather these techniques into a set

\[
Z(y) = \{i \mid 1 \leq i \leq M \land p_i(y) > 0\} . \tag{8.10}
\]
To obtain the total PDF of an output sample \( y \), we marginalize the joint PDF (8.9) over all configurations that could lead to this \( y \):

\[
p(\mathbf{y}) = \sum_{i \in \mathbb{Z}(\mathbf{y})} \left( \prod_{j=1}^{M-1} \int p(x^{i \rightarrow y}, i) \, dx_1 \cdots dx_M \right).
\]

(8.11)

where \( x^{i \rightarrow y} = \{x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_M\} \) is shorthand for the set of candidates with the \( i \)-th candidate fixed to \( y \). The integration is only over the \( M-1 \) candidates that are not fixed.

Although Eq. (8.11) is the PDF of RIS sample \( y \), evaluating this expression in closed form is not feasible, as it involves a high-dimensional integral over a difficult integrand. This would suggest that we could not actually form a primary MC estimator using RIS, as we cannot evaluate its PDF; but evidently, RIS works just fine regardless.

### 8.5.2 Analyzing the RIS Weight

In this section, we will show how RIS can form Monte Carlo estimators despite the problem we demonstrated in the previous section. We begin by rearranging the full RIS estimator from Section 3.1.4.2:

\[
L_{\text{ris}}^{1,M} = f_q(y) \cdot \left( \frac{1}{\hat{p}(y)} \frac{1}{M} \sum_{j=1}^{M} w(x_j) \right) = f_q(y)W(x, z),
\]

(8.12)

where \( W \) is

\[
W(x, z) = \frac{1}{\hat{p}(x_z)} \left[ \frac{1}{M} \sum_{i=1}^{M} w_1(x_i) \right].
\]

(8.13)

From this equation, we can see that Eq. (8.13) forms the contents of the opaque function \( W \) that we used to stand in for the reciprocal PDF of RIS. However, \( W(x, z) \)
is a random variable: For a given output sample \( y \) there are many \( \{x, z\} \) that could have produced it, and which set of values (and therefore, which value for \( W(x, z) \)) is returned by RIS is random.

From context, we can therefore guess that the RIS weight \( W \) is in fact an unbiased estimator of the reciprocal PDF of RIS. Even though the actual PDF (Eq. (8.11)) cannot be computed, RIS provides an estimator of its reciprocal that can be used to form Monte Carlo estimators.

It is worthy of note that so far, we have only guessed at the role of \( W \) from context; Talbot et al. [128] do not derive RIS this way. We will therefore first proceed with a proof that the expected value of \( W \) is indeed the reciprocal PDF of RIS. In the course of this derivation, we will also be able to pinpoint the source of bias in naive reservoir reuse, as well as how to remove it.

### 8.5.3 The Expected RIS Weight

To compute the expected value of \( W \), we need to take a conditional expectation: Given that the output sample is \( y \), what is the average \( W(y) \)? We can do this by taking the expectation of \( W(x, z) \) only over those values of \( x \) and \( z \) for which \( x \rightarrow z = y \), and divide by \( p(y) \): the probability density of the event \( x \rightarrow z = y \). This gives

\[
\mathbb{E}_{x \rightarrow z = y} [W(x, z)] = \frac{\int \cdots \int W(x^{1 \rightarrow y}, i) p(x^{1 \rightarrow y}, i) \, dx_1 \cdots dx_M}{p(y)}, \tag{8.14}
\]

where \( x^{1 \rightarrow y} \) and the integration bounds are the same as in Eq. (8.11). After cancelling the weight sums in the numerator and denominator, expanding Eq. (8.14) yields

\[
\frac{1}{p(y)} \sum_{i \in Z(y)} \int \cdots \int \frac{1}{p(x_i)} \left[ \frac{\sum_{j=1}^{M} w_j(x_j)}{\sum_{j=1}^{M} w_j(x_j)} \right] \left[ \frac{w_i(x_i)}{\sum_{j=1}^{M} w_j(x_j)} \right] \prod_{j=1}^{M} p_j(x_j) \, dx_1 \cdots dx_M. \tag{8.15}
\]
Pulling all terms that do not depend on the integration variables outside results in

\[
\frac{1}{p(y)} \sum_{i \in Z(y)} \frac{p_i(x_i) w_i(x_i)}{\hat{p}(x_i)} \prod_{x_j \in x \setminus x_i} p_j(x_j) \, dx_1 \ldots dx_M. \tag{8.16}
\]

The remaining integral of all candidate PDFs (except \(x_i\), which is fixed to be \(y\)), is simply 1. We can now simplify and use that \(w_i(x) = \hat{p}(x)/p_i(x)\):

\[
= \frac{1}{p(y)} \sum_{i \in Z(y)} \frac{p_i(x_i) w_i(x_i)}{\hat{p}(x_i)} \frac{1}{M} \tag{8.17}
\]

\[
= \frac{1}{p(y)} \sum_{i \in Z(y)} \frac{1}{M} \tag{8.18}
\]

\[
\mathbb{E}_{x_z = y}[W(x, z)] = \frac{1}{p(y)} \frac{|Z(y)|}{M}. \tag{8.19}
\]

This shows two things: If all candidate PDFs are non-zero wherever the target function is non-zero, then \(|Z(y)| = M\), and the RIS weight indeed becomes an unbiased estimator of the inverse RIS PDF. If, however, some of the PDFs are zero for part of the integrand, then \(|Z(y)|/M < 1\), and the inverse PDF is consistently underestimated. This means the expected value is biased to be darker than the true integral.

**A 1D Example.** To demonstrate this, consider the following two candidate PDFs: \(p_1(x) = 1\) and \(p_2(x) = 2H(1/2 - x)\), where \(H(x)\) is the Heaviside step function. The PDFs are illustrated below:

In Fig. 8.6(a), we use these two candidate PDFs to sample a linear ramp, \(\hat{p}(x) = 2 - 2x\), with half the candidates generated from \(p_1\) and the others from \(p_2\), for increasing
Figure 8.6: We show results of RIS for sampling a simple linear target PDF, $\hat{p}(x) = 2 - 2x$. Candidates are produced from a constant PDF ($p_1(x) = 1$) and a step function ($p_2(x) = 2H(1/2 - x)$). We show the inverse PDF of samples produced by RIS, both estimated from the histogram of output samples (dark, thick lines; this is the ground truth), and estimated by the RIS weight (pale lines). The traditional RIS weight (a) is biased where one or more of the PDFs are zero (right half of graph), and the RIS weight (pale lines) does not match the actual distribution of samples (dark lines). Naive unbiased RIS (b) fixes the bias by dividing by the number of non-zero candidate PDFs rather than $M$, but this strategy leads to an extremely noisy RIS weight (c) when a candidate PDF is near-zero rather than zero ($p_2(x) \propto \max(2H(1/2 - x), 10^{-4})$). Our MISed version of the RIS weight (d) is unbiased and robust against small candidate PDFs.
values of $M$. We visualize $1/p(y)$, measured in two different ways: once, by plotting the reciprocal of the histogram of sample locations (solid, dark curves; this is the ground truth), and once as the average of the RIS weight at each location (pale, transparent curves). If standard RIS were truly an estimator of the inverse PDF, these curves should match exactly; however, they do not, and the true reciprocal PDF is instead underestimated.

### 8.5.4 Unbiased RIS

We now show that this bias can be eliminated by modifying $W$: Instead of multiplying by the factor $1/M$, we can choose some (yet unspecified) weight $m(x_z)$:

$$W(x,z) = \frac{1}{\hat{p}(x_z)} \left[ m(x_z) \sum_{i=1}^{M} w_i(x_i) \right]. \quad (8.20)$$

Repeating the derivation of the expected value of $W$ shows that

$$\mathbb{E}_{x_z=y} [W(x,z)] = \frac{1}{\text{pdf}(y)} \sum_{i \in Z(y)} m(x_i), \quad (8.21)$$

indicating an unbiased estimator just requires $\sum_{i \in Z(y)} m(x_i) = 1$.

**Naive approach.** While there are infinitely many ways to choose $m(x)$, the easiest way is to use uniform weights and simply set $m(x_z) = 1/|Z(x_z)|$. That is, instead of dividing by $M$ (the number of candidates), we divide by the number of candidates with non-zero PDFs at that location, creating an unbiased RIS estimator (see Fig. 8.6(b)).

This fixes the bias problem; but, this estimator of the inverse PDF can have problems. Consider a candidate PDF close to, but not exactly, zero such as $p_2(x) \propto \max(H(1/2 - x), 10^{-4})$. As the candidate PDF is never zero, even the original RIS
estimator will be unbiased. However, the estimator of the inverse RIS PDF exhibits high variance, as shown in Fig. 8.6(c).

**Combining with Multiple Importance Sampling.** Luckily, we are able to choose any weights \( m(x_z) \) that sum to 1, for instance:

\[
m(x_z) = \frac{p_z(x_z)}{\sum_{i=1}^{M} p_i(x_z)},
\]

\[(8.22)\]

i.e., the balance heuristic of the candidate PDFs. This solves both bias and noise issues when combining many candidate PDFs using RIS, as shown in Fig. 8.6(d).

**Comparison to RIS.** Talbot et al. [128] propose a different solution for using multiple candidate PDFs in RIS. Where we use \( w_i(x) = \hat{p}(x)/p_i(x) \) (Eq. (8.8)) as the weight, Talbot et al. use \( w_i(x) = \hat{p}(x)/\sum p_i(x) \). By replacing the individual PDFs by a single average PDF, Talbot forgo noise and bias issues that arise when mixing multiple candidate PDFs. In addition, if the sum of candidate PDFs is closer to the target distribution than the individual PDFs, then Talbot et al.’s approach may further reduce noise compared to ours. However, there is a crucial difference between the two approaches: Talbot et al. evaluate all PDFs for each candidate sample; if each candidate sample uses a different PDF, then the cost of their approach is \( O(M^2) \) PDF evaluations. In contrast, our approach evaluates only one PDF for each candidate, and all PDFs only once more when computing the final MIS weight (Eq. (8.22)), equivalent to a cost of \( O(M) \). This is especially crucial in our case, in which evaluating the PDF may involve tracing a ray; the quadratic cost of Talbot et al.’s approach then makes it completely infeasible in this use case, whereas the linear cost of our approach offers unbiasedness at affordable cost.
Algorithm 7: Unbiased combination of multiple reservoirs.

**Input:** Reservoirs \( r_1 \) and the pixels \( q_i \) they originated from.

**Output:** An unbiased combination of the input reservoirs.

```algorithm
function combineReservoirsUnbiased(q, r_1, r_2, ..., r_k, q_1, ..., q_k)
    Reservoir s
    foreach \( r \in \{r_1, ..., r_k\} \) do
        s.update(\( r.y \), \( \hat{p}_q(r.y) \cdot r.W(r.y) \cdot r.M \))
        s.M \leftarrow r_1.M + r_2.M + ... + r_k.M
    Z \leftarrow 0
    foreach \( q_i \in \{q_1, ..., q_k\} \) do
        if \( \hat{p}_{q_i}(s.y) > 0 \) then
            Z \leftarrow Z + r_i.M
        m \leftarrow 1/Z
    s.W \leftarrow \frac{1}{\hat{p}_q(s.y)} (m \cdot s.w_{\text{sum}}) \quad // \text{Equation (8.20)}
    return s
```

8.5.5 A Practical Algorithm for Unbiased Reuse

We can now apply our bias correction to our algorithm for sample reuse (Alg. 6). The bias is introduced when combining multiple reservoirs (Alg. 5): a pixel \( q \) gathers reservoirs \( r_i \) from its neighboring pixels, each of which contributes a sample \( r_i.y \); however, the PDF of this sample may be zero where the integrand at \( q \) is not. For example, candidates that lie below the hemisphere are normally discarded. However, neighboring pixels may have differently oriented surface normals, and may discard samples that would have non-zero contribution at \( q \). Similarly, our algorithm discards samples that are occluded after the first round of resampling (effectively setting the PDF to zero); however, a sample occluded at one pixel may be visible at its neighbor, and discarding it causes bias.

Each sample \( r_i.y \) is the result of resampling, and we do not know its true PDF (since Equation (8.11) cannot be evaluated in closed form). However, as long as we know an approximate form of this PDF that is zero whenever the real PDF is zero, we can use it instead to compute an unbiased weight. For pixel \( q_L \), we use \( \hat{p}_{q_L}(x) \) as
8.6 Design and Implementation Choices

We implemented both biased and unbiased variants of our algorithm in a GPU-based real-time rendering system. We have made various design choices to improve robustness and performance, as well as to limit the impact of bias, which we detail in this section. We also specify the parameters used in our implementation. In general our unbiased algorithm is computationally more expensive, and we choose different parameters for our biased and unbiased variants such that they have approximately equal cost.

Candidate Generation. We sample $M = 32$ initial candidates by importance sampling emissive triangles based on their power, and then uniformly generate a point $x$ on the selected triangle (i.e. $p(x) \propto L_e(x)$). If an environment map is present in the
scene, 25% of candidates are instead generated by importance sampling the environment map. Importance sampling for both triangles and environment map locations is accelerated using an alias table \[138\]. We also experimented with pregenerating a list of VPLs on emissive triangles. Doing so yields higher performance at the cost of some visual artifacts, and may be an option for real-time applications with limited render-times. It would also be possible to use higher quality samples as initial candidates—such as those produced by the data structure of Moreau et al. \[103\]—but this proved to significantly increase runtime in our preliminary tests.

**Target Distribution.** At each resampling step in our algorithm, we weight samples based on a target PDF. We use the unshadowed contribution \(\hat{p}_q(x) = \rho(y, \bar{y}, \omega)L_e(x, \bar{x})G(y, x)\) as the target distribution at each pixel. We use a unified material model for all geometry in the scene, consisting of a dielectric GGX microfacet layer atop a diffuse Lambertian substrate. If more sophisticated material models are used and evaluating the BSDF for each candidate is too expensive, approximations to the BSDF may be used.

**Neighbor selection.** For spatial reuse, we found that deterministically selected neighbors (e.g. in a small box around the current pixel) lead to distracting artifacts, and we instead sample \(k = 5\) \((k = 3\) for our unbiased algorithm) random points in a 30-pixel radius around the current pixel, sampled from a low-discrepancy sequence. As an alternative, using a hierarchical À-Trous sampling scheme \[31, 120\] also produced promising results, at the cost of some artifacts, and may be interesting for future work. For temporal reuse, we compute motion vectors to project the current pixel’s position into the previous frame, and use the pixel there for temporal reuse.

For our biased algorithm, reusing candidates from neighboring pixels with substantially different geometry/material leads to increased bias, and we use a simple heuristic to reject such pixels: we compare the difference in camera distance, and
the angle between normals of the current pixel to the neighboring pixel, and reject
the neighbor if either exceed some threshold (10% of current pixels depth and 25°,
respectively). This strategy is similar to those used in selective blurs for real-time
denoising, and we found it to substantially reduce bias. We use \( n = 2 \) (\( n = 1 \) for our
unbiased algorithm) spatial reuse passes.

**Evaluated Sample Count.** Our Alg. 6 assumes \( N = 1 \), i.e. a single sample is eval-
uated at the end of the frame. A secondary estimator that averages \( N \) evaluations
of our algorithm can be trivially accomodated by storing \( N \) independent reservoirs
at each pixel, and performing reuse on all reservoirs in parallel. For our unbiased
algorithm, we use \( N = 1 \) for interactive frame-rates; our biased algorithm uses \( N = 4 \)
instead. For non-interactive render times, we simply average images of independent
executions of our algorithm.

**Reservoir storage and temporal weighting.** At each pixel, we only store the infor-
mation of the pixel’s reservoir: The selected sample \( y \), the number of candidates \( M \)
that contributed to the pixel, and the estimate of the reciprocal PDF \( W \). For \( N > 1 \),
we store multiple samples \( y \) and weights \( W \) at each pixel to accomodate multiple
reservoirs. With temporal reuse, the number of candidates \( M \) contributing to the
pixel can in theory grow unbounded, as each frame always combines its reservoir
with the previous frame’s. This causes (potentially stale) temporal samples to be
weighted disproportionately high during resampling. To fix this, we simply clamp
the previous frame’s \( M \) to at most \( 20 \times \) of the current frame’s reservoir’s \( M \), which
both stops unbounded growth of \( M \) and bounds the influence of temporal informa-
tion.
Figure 8.7: Comparison of roughly equal-time renderings of a streaming implementation of Talbot et al. [128] with our biased and unbiased spatiotemporal sample reuse. A converged reference is also shown for comparison. BISTRO has 20,638 emissive triangles and an environment map, BURGER RESTAURANT has 7,517 textured emissive triangles and a mostly-occluded environment map, SUBWAY has 23,452 textured emissive triangles, and ZERO DAY animation has 10,973 dynamic emissive triangles.
8.7 Results

We prototyped our method in the open-source Falcor rendering framework [6] to take advantage of hardware-accelerated ray tracing. We call our algorithm Reservoir-based Spatio-Temporal Importance Resampling, or ReSTIR for short. We tested our technique on various scenes containing thousands to millions of emissive triangles. Renderings and timings were obtained on a GeForce RTX 2080 Ti GPU, except for the Amusement Park scene, which required use of a Titan RTX due to high memory requirements.

The render times that we report include the cost of sample generation, ray tracing and shading. We do not include G-buffer rasterization cost, as this is shared between all rendering methods (and averages 1-2 ms). We report image errors of each method compared to an unbiased reference rendered at high sample count. Errors are reported as Relative Mean Absolute Error (RMAE), which we found less sensitive to isolated outliers than mean squared error (MSE).

For methods using temporal reuse, our figures show the final frame in a 20 frame animation involving fast camera movement. This avoids the lower quality expected during any warm up period without providing any artificial advantage by temporally supersampling a single view. Each frame in the sequence uses the same computation budget as the final frame.

Figure 8.7 compares the biased and unbiased versions of our spatiotemporal reuse with RIS [128] at equal time. To allow for a fair baseline comparison, we compare against our streaming version of RIS (reservoir resampling), as we found it consistently faster (20%-30% speedup) than non-streaming implementations. Our methods employing spatial and temporal reuse significantly outperform RIS without reuse, both visually and in terms of error. In some scenes (e.g. Subway), the baseline image is barely recognizable, but our spatiotemporal reuse image is nearly converged. In
all scenes, our biased method has considerably less variance, at the cost of some energy loss and image darkening. The energy loss is most pronounced in regions with difficult lighting, e.g. shadow boundaries, sharp highlights and complex geometry such as trees.

Figure 8.8 shows equal-time comparisons of our biased and unbiased spatiotemporal reuse versus a state-of-the-art real-time light sampling technique of Moreau et al. [103] based on a bounding volume hierarchy (BVH) over light sources. Our technique has substantially lower error than Moreau et al.’s BVH-based approach. We found that the light BVH generally under-performs even our reservoir resampling algorithm (without reuse); in all further results we use streaming RIS as the baseline for comparisons.

Figure 8.9 show plots of how RMAE evolves with increased render time for six different methods: sampling lights according to power and then applying MIS [134] with BSDF and area-weighted sampling; Moreau et al. [103]’s light BVH; reservoir resampling, as well as three versions of our algorithms: biased and unbiased spatiotemporal reuse, as well as biased spatial reuse without temporal reuse. The last variant makes it possible to evaluate our algorithm for still images. In all scenes, our biased spatiotemporal reuse has the lowest error at interactive render times, usually by a significant margin. However, as render time increases, the error due to bias
Figure 8.9: The evolution of error (relative mean absolute error) in our scenes over render time. We compare Veach et al.-style MIS with lights sampled according to power, Moreau et al.’s light BVH, a streaming implementation of Talbot et al.’s RIS (i.e. reservoir resampling), and three variants of our algorithm: Biased and unbiased spatiotemporal and visibility reuse; as well as a biased form of spatial and visibility reuse, with no reliance on temporal information.
dominates, so our unbiased spatiotemporal reuse eventually exhibits lower error (usually at around 1 s). In most scenes, biased spatial reuse also offers competitive performance without relying on knowledge from prior frames. The lack of temporal history also limits propagation of bias across multiple frames, and at longer render times this method can overtake biased spatiotemporal reuse due to reduced bias. In all scenes, we significantly outperform prior work.

To demonstrate the performance of our method at non-interactive render times, we compare reservoir resampling and our methods on the AMUSEMENT PARK scene at 1 s render time in Figure 8.10. Even at comparatively high render times, we still significantly outperform the baseline. Our biased spatiotemporal reuse is nearly noise-free, but the bias is apparent; if problematic, unbiased spatiotemporal reuse offers similar performance with slightly higher variance.

### 8.8 Discussion

We conclude this chapter with a short summary of our contributions, and follow with a review of the limitations of our proposed algorithm.
8.8.1 Contributions

In this chapter, we have presented the following contributions:

• We introduced a new Monte Carlo approach based on reservoir resampling and reservoir reuse that allows approximately sampling challenging distributions. We achieved this by spatially and temporally reusing samples in reservoirs.

• We analyzed the bias of this algorithm, and introduce new analysis of the RIS estimator that allowed us to derive an unbiased rendering algorithm, at the cost of extra computation.

• We demonstrated an implementation of our algorithm on contemporary graphics hardware, and deliver one to two orders of magnitude reduction in error compared to previous approaches on scenes with thousands to millions of dynamic light sources. Our approach requires only simple image-sized data structures and is well suited to current graphics hardware.

One way to view our technique is that we have shown that filtering and denoising need not remain a post-process that is performed once rendering completes—effectively, we have moved denoising into the core of the renderer and filter PDFs rather than colors. We see this as an important insight to spur future development of denoising algorithms, which have thus far remained specialized (and often carefully hand-tuned) postprocesses. It may also be worthwhile to develop new post-process denoising approaches that are adapted to the characteristics of the output of our algorithm or make use of unique features that it can provide, such as the individual candidate visibility values.
8.8.2 Limitations

Similar to other algorithms relying on sample reuse, our method relies on exploiting correlations between pixels to improve image quality. When such opportunities are not available—e.g., near disocclusions, lighting discontinuities, high geometric complexity, fast moving lights—the quality of our method degrades and the noise reduction compared to the input samples is modest. While we generally saw our method performing better than prior work even in such challenging cases, making our method more robust to cases in which reuse is not possible is a fruitful direction for future work. Unlike post-processing methods such as denoising, our method still has the opportunity to trace additional samples, and it would be interesting to explore metrics that determine where our method fails, and allocate additional samples to those regions.

The main data structure of our algorithm consists of image buffers. While this makes our method fast, simple and memory efficient, it limits the use of our method to operations on the first vertex of the camera path (i.e., the primary hit point), and it cannot be easily extended to direct lighting or global illumination beyond the first hit. While direct lighting at the primary hit is an important problem in interactive applications, extending our algorithm beyond screen-space is an important area for improvement, and we introduce an extension of our algorithm in Chapter 9 which operates in scene-space instead.

While the analysis in Section 8.5 was successful in deriving an approach for removing bias from ReSTIR, our proposed method is suboptimal when combining reservoirs with significantly different distributions. This manifests as increased variance in glossy reflections and on surfaces close to light sources. This is partially caused by the fact that our method compensates for suboptimal sampling with a single weight multiplied into $W$. In Chapter 9, we will introduce a more flexible approach that
retains the ability to remove bias while improving variance in neighborhoods with rapidly changing sample distributions.

### 8.8.3 Follow-up Work

Since the publication of this work, several related works have been published that extend or use some of the ideas proposed in this chapter.

Although the goal of this chapter was to introduce an efficient algorithm for real-time direct lighting, there are still many opportunities for performance optimization in the implementation of our algorithm. Wyman et al. [145] demonstrate a highly optimized implementation of ReSTIR that is up to $7 \times$ faster than the implementation we show in this chapter. They also discuss a number of tradeoffs between bias and performance that allow greater flexibility in real-time use cases where computational budget is limited.

Boksansky et al. [16] propose a grid-based variant of ReSTIR. First, a uniform scene-space grid is built, where each grid cell contains a list of reservoirs. The reservoirs in each cell are then populated using an initial round of RIS with a simple target function proportional to light intensity and inverse squared distance to the cell. The rest of the algorithm proceeds identical to ReSTIR, except that the reservoirs at each pixel are populated via resampling lights from the nearest grid cell, instead of from an initial round of RIS as in ReSTIR. This can be viewed as a form of culling, where a crude target function evaluated at coarse locations produces a smaller set of important lights, which are subsequently fed into standard ReSTIR with a more complex target function evaluated at each pixel. In the next chapter, we introduce a related idea based on an octree data structure, except samples are fed back and forth from screen-space and scene-space data structures.

Lin et al. [92] and Ouyang et al. [107] both propose extensions of ReSTIR beyond direct lighting. Lin et al. focus on transport in participating media, whereas Ouyang
et al. treat the surface case. In both cases, reuse of samples must be modified to account for the higher-dimensional problem space. Ouyang et al. choose to treat the third vertex of arbitrary length paths as a point light source, which allows for similar reuse strategies to be applied as those discussed earlier in this chapter. Lin et al. propose an additional strategy, in which directions instead of the vertex location are reused. In both cases, reservoirs are still stored on pixels. It would be interesting to combine these algorithms with the octree-based approach proposed in the next chapter.
The algorithm we introduced in the previous chapter—ReSTIR—used RIS as a mechanism for reusing sampling efforts across pixels to improve the distribution of samples used for rendering. While we showed an application that improved direct lighting, the basis of it is a general mechanism that could be used to derive new sampling algorithms in other problem domains.

In this chapter, we will introduce two extensions to this mechanism, both to make it more robust and to improve its generality. We begin with a new method for performing robust combinations of multiple reservoirs in Section 9.1. The debiasing and MIS method we introduced in Chapter 8 was effective at removing bias, but is not ideal when it comes to preventing variance from combinations of dissimilar reservoirs. We will fix this issue by first showing a new derivation of RIS from a secondary estimator, and then using this as a basis for a new weighting scheme that prevents variance much more robustly.

Second, we propose a generalization of ReSTIR in Section 9.2 that shows how it can be applied to domains other than screen-space. The implementation we presented in the previous chapter stored and reused reservoirs at pixel locations; although this proved very effective for the problem domain of direct lighting, it is difficult to apply the same algorithm to other problems. We show how ReSTIR can be generalized to operate in scene-space instead, where reservoirs are stored at locations in the scene. This requires aggregating multiple estimation locations into a single aggregate reservoirs, on which spatial and temporal reuse are performed.
Finally, we demonstrate and analyze two more deficiencies in reservoir reuse in Section 9.3 and propose potential solutions. Although these solutions do not lead to consistent improvement over the baseline, they point to fruitful directions for future work.

### 9.1 Pairwise MIS for Robust Reservoir Reuse

One of the major challenges of reservoir reuse is that a large number of samples, each from a different distribution, are combined and resampled into a single reservoir. Without careful weighting of the samples, these combinations are likely to result in noise or even bias. In Chapter 8, we introduced an analysis of bias in the RIS estimator and showed how it could be prevented using a modified estimator $W$ for the reciprocal PDF. However, as we will demonstrate in this section, the robustness of this estimator in the presence of diverse sample distributions is limited. Additionally, this estimator requires two passes over the input reservoirs, which adds computational cost on graphics hardware due to the need to store the inputs.

In this section, we introduce a new form of MIS for the RIS estimator that we term pairwise MIS. We begin in Section 9.1.1 with a novel derivation of the RIS estimator that provides new insights and allows us to formulate MIS based estimators (Section 9.1.2) that differ from the methods introduced in Chapter 8. We show a new MIS heuristic tailored to our use case (Section 9.1.3) that is both more effective and more efficient than the method introduced in Chapter 8, and allows for the effective inclusion of BSDF samples. We evaluate our new method in Section 9.1.4.
9.1.1 RIS as a Secondary MC Estimator

In this section, we will rederive RIS as a special form of secondary MC estimator. This alternate derivation of RIS will make it obvious how the weighting terms (i.e. \( W \)) come to be, and will allow us to derive modified RIS estimators that will allow for more flexible MIS techniques.

Recall the standard secondary estimator (Eq. (3.2)) in Monte Carlo:

\[
I^{(k)} = \frac{1}{k} \sum_{i=1}^{k} \frac{f(x_i)}{p(x_i)}. \tag{9.1}
\]

In a scenario where evaluating \( f \) is expensive, we may not be able to afford large \( k \). Instead of limiting the value of \( k \), we could alternatively allow arbitrarily large values for \( k \) while only selecting one of the \( k \) samples to evaluate. Say we choose sample \( x_i \) with probability proportional to some weight \( w_i \). The sum collapses, and a new term appears to account for the probability of the output sample being chosen; the estimator then becomes

\[
I^{(k)} = \frac{1}{k} \frac{f(x_z)}{p(x_z)} \cdot \frac{1}{p(z|x)}, \quad \text{where} \quad p(z|x) = \frac{w_z}{\sum_{i=1}^{k} w_i}. \tag{9.2}
\]

If we choose the weights as \( w_i = \hat{p}(x_i)/p(x_i) \) for some arbitrary function \( \hat{p} \) and rearrange, we obtain

\[
I^{(k)} = \frac{f(x_z)}{k \hat{p}(x_z)} \cdot \frac{1}{w_z} \sum_{i=1}^{k} w_i \tag{9.3}
\]

\[
= \frac{f(x_z)}{p(x_z)} \cdot \left( \frac{p(x_z)}{\hat{p}(x_z)} \frac{1}{k} \sum_{i=1}^{k} w_i \right) \tag{9.4}
\]

\[
= f(x_z) \frac{1}{\hat{p}(x_z)} \cdot \left( \frac{1}{k} \sum_{i=1}^{k} w_i \right). \tag{9.5}
\]
Eq. (9.5) is identical to the original RIS estimator (Eq. (3.11)). We can see that RIS is mathematically identical to a standard secondary MC estimator, where only one of the samples is randomly selected to be evaluated. The RIS weight then arises simply from dividing by the selection probability and rearranging.

### 9.1.2 A Multiple Importance Sampled RIS Estimator

This reformulation allows us to plug in any modified secondary estimator, and derive the corresponding RIS estimator that has the correct weights. For example, we could allow each sample $x_i$ to follow a different distribution $p_i(x_i)$, similar to Section 8.5.1. More interestingly, we could begin with the Multiple Importance Sampled version (Section 3.1.5) instead of the unweighted secondary estimator:

$$I^{(k)} = \frac{1}{k} \sum_{i=1}^{k} m_i(x_i) \frac{f(x_i)}{p_i(x_i)}. \quad (9.6)$$

This setup allows us to choose any MIS weight $m$ to combine the $k$ different sampling techniques (or candidate distributions in the RIS view), and leaves us significant flexibility. From this equation, we can derive two distinct RIS estimators: If we were to use the same selection weights ($w_i = \hat{p}(x_i)/p_i(x_i)$) as in standard RIS, we obtain

$$I^{(k)} = f(x_z) \frac{m_z(x_z)}{\hat{p}(x_z)} \cdot \left( \frac{1}{k} \sum_{i=1}^{k} w_i \right). \quad (9.7)$$

This naive estimator is not very useful and does not provide any variance reduction. In fact, it does the opposite: This estimator is nearly identical to the standard RIS estimator, except for an additional factor ($m$) not sampled by $\hat{p}$, which adds additional variance.

Fortunately, the selection weights in RIS can be chosen freely, and a much more useful formulation can be obtained if we instead incorporate the MIS weight into the

---

**9.1 PAIRWISE MIS FOR ROBUST RESERVOIR REUSE**

---

221
selection weights, \( w_i = m_i(x_i) \hat{p}(x_i)/p_i(x_i) \). With this, we obtain the modified RIS estimator:

\[
I^{(k)} = f(x_z) \frac{1}{\hat{p}(x_z)} \cdot \left( \frac{1}{k} \sum_{i=1}^{k} w_i \right). \tag{9.8}
\]

This estimator accomplishes two things: Because the MIS weight is incorporated into the selection weight, RIS preferentially selects samples that are sampled well by their corresponding candidate distribution, leading to improved convergence toward the target distribution. Simultaneously, the presence of the MIS weight in the selection weight can lead to significant reduction in the variance of the estimator of the reciprocal PDF, \( W \). If, for a particular sample \( x_i \), its distribution substantially undersamples the target distribution, then the selection weight \( \hat{p}(x_i)/p_i(x_i) \) can introduce significant variance into the estimator \( W \). The MIS weighted estimator we introduced in the previous chapter (Eq. (8.20)) can only insufficiently handle this case, as it only allows for a single MIS weight for the entire estimate of \( W \). In contrast, the estimator we propose in Eq. (9.8) has a different MIS weight for each term \( w_i \) and allows other candidate distributions to compensate if any one distribution fails.

### 9.1.3 An efficient MIS heuristic

The RIS estimator we described in the previous selection provides a foundation on which to build robust estimators with MIS. However, it does not prescribe any particular MIS heuristic. The choice of heuristic impacts both the variance and cost of the estimator, and is crucial to obtain an efficient rendering algorithm.

For example, an obvious choice is to use the balance heuristic

\[
m_i(x_i) = \frac{p_i(x_i)}{\frac{1}{k} \sum_{j=1}^{k} p_j(x_i)}. \tag{9.9}
\]
While this heuristic is a provably good choice in terms of variance reduction [134], in our case it also comes at high computational cost. Because the MIS weight is evaluated for each of the $k$ candidates, and each evaluation involves $k$ terms, the cost of the resulting RIS estimator would grow quadratically in $k$, i.e. $O(k^2)$. The algorithms we introduce later in this chapter (Section 9.2, Section 9.3.3) perform combinations of large numbers (potentially thousands) of reservoirs and quadratic growth would be unsustainable.

The reason the balance heuristic in Eq. (9.9) incorporates a sum over the PDFs of all sampling techniques is that at sample $x_i$, any one technique could be degenerate and be responsible for a large amount of variance. This defensive approach maximizes robustness, but is responsible for the high computational cost. A key observation for the case of reservoir reuse is that there usually exists a canonical technique which is likely to behave reasonably well (i.e. is not degenerate) over the entire domain.

For example, in spatial reuse we combine the reservoir at a pixel $q$ with reservoirs from randomly selected neighboring pixels. While the distribution of samples from neighboring pixels may be poor for the target distribution at $q$ and introduce variance if used directly, the sample at the reservoir $q$ itself from the previous stage of reuse has undergone RIS at least once to approximate $\hat{p}_q$, and is therefore a reasonable sample for the target distribution $\hat{p}_q$ in subsequent stages of resampling.

Given such a canonical technique, we propose to use an alternative pairwise form of MIS that corresponds to using the balance heuristic between a pair of two techniques: The technique that produced the sample, and the canonical technique $c$:

$$m_i(x_i) = \begin{cases} \frac{p_i(x_i)}{p_i(x_i) + p_c(x_i)} & \text{if } i \neq c \\ 1 + \sum_{i \neq c} 1 - m_i(x_c) & \text{if } i = c \end{cases} \quad (9.10)$$

Because we combine pairs of techniques using the balance heuristic, the variance proof of Veach et al. [134] promises that the variance of this estimator will never be
much worse than the canonical technique in isolation, and usually better if a non-canonical technique is more likely to produce a sample $x_i$. This means that as long as the canonical technique does not itself cause high variance, this combination is robust. However, the average cost of evaluating this heuristic is constant, regardless of $k$.

**Sample Counts and Overcounting.** In Eq. (9.10), we implicitly assumed that all techniques have the same number of samples associated with them. To account for the effective number of candidates that contributed to different reservoirs, we additionally weight techniques based on their effective $M$ as described in the previous chapter: Each PDF $p_i$ in Eq. (9.10) is replaced by $M_i \cdot p_i$. We also observe that the canonical technique $p_c$ appears in $k$ pairwise combinations and is therefore overrepresented. Even if the PDF of all techniques was identical, the canonical technique $c$ would receive $k$ times the weight of all other techniques. To account for this, we divide the effective $M$ of the canonical technique by $k$ to compensate for the overcounting, and replace $p_c$ in Eq. (9.10) with $M_c / k \cdot p_c$.

**Discussion.** We show an outline of the proposed RIS algorithm in Alg. 8, which combines multiple reservoirs using pairwise MIS. This algorithm has several interesting properties: First, similar to Alg. 7 from Section 8.5.5, it completes in time $O(k)$, in contrast to the quadratic cost of the naive balance heuristic. Second, this algorithm only iterates *once* over the input reservoirs, where the MIS strategy from the previous chapter required doing so twice. This may seem insignificant, but it constitutes a significant difference in practice: If the input reservoirs are generated sequentially (e.g. by randomly choosing a sequence of neighboring pixels), Alg. 8 can receive the reservoirs in a streaming fashion, immediately discarding each input reservoir after processing it. In contrast, Alg. 7 from Chapter 8 requires two passes over the input, once to select the output sample and once more to compute the MIS weight.
Algorithm 8: Combination of multiple reservoirs using pairwise MIS.

**Input**: Reservoirs $r_i$ and the pixels $q_i$ they originated from; canonical reservoir $r_c$ and pixel $q_c$.

**Output**: An unbiased combination of the input reservoirs using pairwise MIS.

```python
def combineReservoirsPairwise(q_c, r_c, r_1, r_2, ..., r_k, q_1, ..., q_k):
    Reservoir s
    m_c ← 1
    foreach $r_i$ ∈ {r_1, ..., r_k} do
        m_i ← $r_i$.M · p_i(r_i.y) / ($r_i$.M · p_i(r_i.y) + $r_c$.M/k · p_c(r_c.y))
        m_c ← m_c + (1 - $r_c$.M · p_c(r_c.y) / ($r_i$.M · p_i(r_i.y) + $r_c$.M/k · p_c(r_c.y)))
        s.update(r_i.y, $p_q$(r_i.y) · r_i.W(r_i.y) · m_i)
        s.update(r_c.y, $p_q$(r_c.y) · r_c.W(r_c.y) · m_c)
    s.M ← $r_c$.M + r_1.M + r_2.M + ... + r_k.M
    s.W(s.y) = 1 / $p_q$(s.y) · (1 / (1 + k · s.w_sum))
    return s
```

This requires storing all input reservoirs, which constitutes a significant performance difference on graphics hardware.

### 9.1.4 Results

We implemented our proposed pairwise MIS on top of the ReSTIR algorithm introduced in Chapter 8, and test it on the same scenes as in the evaluation of ReSTIR. We show both equal-time renderings and RMAE graphs of the ReSTIR baseline and the modified version of ReSTIR with pairwise MIS added in Fig. 9.1. We use the biased version of ReSTIR, as it provided the largest speedup over prior work at real-time rates. Even starting from this already efficient baseline, pairwise MIS leads to significantly reduced noise in glossy reflections without added cost, both visually and in terms of RMAE.

**Incorporating BSDF Sampling.** The ReSTIR algorithm from the previous chapter generated candidates by uniform sampling of the area of all emitters in the scene.
Figure 9.1: We compare roughly equal time renderings of our a ReSTIR baseline to a modified version of ReSTIR that uses pairwise MIS. We show both algorithms with and without added BSDF samples (denoted with $+\rho$). In both scenarios, pairwise MIS leads to considerable improvements, both visually and in terms of RMAE.
For points close to light sources or in glossy reflections, this distribution can be sufficiently far from the integrand that even RIS with reuse can struggle to retrieve useful samples. We opt to additionally add 1 sample generated by sampling the BSDF to the candidates in ReSTIR, and show results both with and without BSDF sampling (algorithms with BSDF sampling are denoted by $+\rho$). BSDF samples alone provide moderate to absent improvement in RMAE in ReSTIR, but their benefit is compounded with pairwise MIS, and their combination can lead to significant variance reductions at comparable render times to ReSTIR.

### 9.2 Scene-Space Reservoir Reuse

The algorithm we introduced in Chapter 8 operated entirely in *screen-space*, wherein all reservoirs are stored at pixel locations in a screen-sized array. While this has several advantages, such as simplicity of data structures and ease of finding neighboring
reservoirs, it also limits the generality of the algorithm. For example, only the parts of
the scene directly visible to the sensor can partake in reservoir reuse, restricting the
algorithm to paths of length 2. Additionally, the density of reservoirs is directly tied
to the screen resolution, even though it may be desirable to adapt the distribution of
reservoirs e.g. based on the frequency of the lighting, or based on the capability of
the underlying hardware. Lastly, pixel-sized geometric detail such as foliage make it
challenging to perform temporal reuse. Even with backprojection to compensate for
camera motion, aliasing from small geometry can cause a pixel to see entirely dif-
f erent scene locations across subsequent frames in an animation, which introduces
significant variance on fine geometric detail.

In this section we will introduce a new variant of ReSTIR in which reservoirs are
stored at locations in scene-space, i.e. on subsets of Θ instead of at pixel locations.
In this algorithm, each reservoir now occupies a finite region of space, and there
is no longer a unique scene location that determines the integrand (and thus the
target distribution) for the reservoir. This poses several new challenges of how these
reservoirs should be represented, how their target distribution should be chosen, and
what data structure to store them in. We begin with an overview of the proposed
algorithm in Section 9.2.1 and proceed to address these issues in order in sections
Section 9.2.3 and Section 9.2.4. Finally, we compare the resulting algorithm to ReSTIR
in Section 9.2.5.

9.2.1 Algorithm Overview

We present an overview of our algorithm in Alg. 9. Our algorithm proceeds in several
phases, which we illustrate in Fig. 9.3 and explain in more detail below. The goal
of the algorithm is to estimate direct lighting at each vertex of each sensor path
generated in the current frame.
Algorithm 9: Our algorithm for scene-space ReSTIR.

Input: Data structure containing the previous frame’s reservoirs
Output: The current frame’s reservoirs

function sceneSpaceReSTIR(prevFrameDataStructure)

dataStructure ← new DataStructure()

// Path generation phase
foreach pixel q ∈ Image do
  x = samplePath(q)
  foreach x_i ∈ x do
    Reservoir r ← RIS(x_i) // Alg. 4
    generate z ∼ p
    dataStructure.recordReservoir(x_i, r, z)

// Aggregation phase
foreach spatialBucket ∈ dataStructure do
  Reservoir s = new Reservoir()
  r_1, r_2, ..., r_k ← spatialBucket.reservoirs
  z_1, z_2, ..., z_k ← spatialBucket.auxiliarySamples
  // Determine target distribution (Section 9.2.2)
  s.hat.p = aggregateTargetDistributions(r_1.hat.p, ..., r_k.hat.p)
  s.update(z_1, ..., z_l) // Generate canonical sample (Section 9.2.3)

  // Temporal reuse
  prevBucket = prevFrameDataStructure.findBucket(spatialBucket.center)
  s ← combineReservoirsPairwise(s, prevBucket.s)

  // Incorporate the current frame’s reservoirs
  s ← combineReservoirsPairwise(s, r_1, r_2, ..., r_k)

  // Visibility reuse
  if shadowed(s.y) then
    s.W ← 0
  // Store new reservoir in data structure
  spatialBucket.s ← s

// Spatial reuse
foreach spatialBucket ∈ dataStructure do
  Q ← pickSpatialNeighbors(spatialBucket.center)
  R ← {dataStructure.findBucket(x).s | x ∈ Q}
  spatialBucket.s ← combineReservoirsPairwise(spatialBucket.s, R)

// Compute pixel color
foreach pixel q ∈ Image do
  foreach x_i ∈ x_q do
    spatialBucket ← dataStructure.findBucket(x_i)
    Reservoir r ← spatialBucket.retrieveInput(q, i)
    Reservoir s ← spatialBucket.s
    r ← combineReservoirsPairwise(r, s)
  Image[q] ← Image[q] + shadePixel(r, q)

return reservoirs
Figure 9.3: Our proposed scene-space extension of ReSTIR proceeds in several phases. We first generate paths for each pixel in the image (a). Groups of path vertices are then aggregated into spatial buckets based on a spatial data structure (b). Each bucket is assigned an aggregate reservoir, which is populated by samples from each path vertex (c). Spatiotemporal reuse is then performed across the aggregate reservoirs (d); the resulting samples in the aggregate reservoir are reinjected into the reservoir at each vertex, and the resulting samples are shaded (e).

We accomplish this by grouping nearby vertices into an aggregate reservoir that tries to distribute samples according to their average contribution over all path vertices that land in the bounds of the aggregate. At the start of the frame, each aggregate reservoir is populated with samples from all path vertices in the aggregate; temporal and spatial reuse is then performed over aggregate reservoirs only, before transferring samples from the aggregate reservoirs back to the path vertices.

The aggregate reservoirs are stored in a spatial data structure that adaptively divides space into a series of non-overlapping buckets that represent a small volume of space; all path vertices that land within the volume of a bucket form one aggregate. We detail our choice of data structure in Section 9.2.4.
Path generation phase. We begin each frame by generating a transport path $\mathbf{x}$ for each pixel in the image. This path could be of length 1 as in ReSTIR, or longer to account for direct lighting at indirectly visible vertices. We initialize a reservoir $r$ at each path vertex of said path with some initial set of $M$ candidates. For reasons we explain in Section 9.2.3, we additionally generate a single auxiliary sample $z$ distributed uniformly over the area of all emitters. The reservoir $r$ and the uniform sample $z$ are inserted into the spatial data structure to be recorded in the nearest bucket for the next phase.

Aggregation phase. After the data structure is populated, we create an aggregate reservoir $s$ in each bucket and populate it. We first retrieve the initial reservoirs $r_i$ and uniform samples $z_i$ of the path vertices that are contained in this bucket, and then compute an aggregate target distribution (Section 9.2.2) that mimics the average response of all target distributions of the initial reservoirs $r_i$. Using this target distribution, $s$ is populated by the uniform samples $z_i$ (Section 9.2.3). We then do temporal reuse by retrieving the closest bucket in the previous frame’s data structure, and performing pairwise reuse between $s$ and the previous frame’s aggregate reservoir. Finally, we combine $s$ with the initial reservoirs $r_i$ using pairwise reuse to form the fully populated aggregate reservoir.

Spatial reuse phase. We proceed to combine each aggregate reservoir with $k$ randomly selected neighboring reservoirs in the current frame, similar to spatial reuse in the screen-space algorithm.

Shading phase. Finally, we iterate through the vertices of all paths generated in this frame, and recombine the aggregate reservoirs with the reservoir at each vertex. Given the vertex location $x_i$, we retrieve the closest bucket from the spatial data struc-
ture and perform pairwise reuse with its aggregate reservoir. The resulting output sample is used to estimate direct lighting at \( x_i \) and is recorded in the image.

### 9.2.2 Aggregate Target Distributions

Much like the reservoirs we handled in Chapter 8, the choice of target distribution for aggregate reservoirs is crucial to their efficiency. However, where before we could obtain reasonable target distributions by selecting a subset of terms from the integrand of interest, there is now a collection of integrands that should (ideally) all be importance sampled by a single target distribution. In addition, the target distribution must also be efficient to evaluate and be compact in storage, as it is evaluated frequently and stored for each aggregate reservoir.

This rules out naive approaches, such as tabulating the average of all target distributions, or evaluating all target distributions on-the-fly and taking their mean. However, we observe that all of the integrands in the aggregate still consist of a product of terms, and we can construct a new target distribution that represents the product of the analytic (but approximate) average of each of the terms over the aggregate.
We base the aggregate distribution on the target distribution
\[ p_q(y) = \rho(x, \bar{x}, \omega) L_e(y, \bar{y}) G(x, y) \]
we used in Chapter 8, except where the terms are now averaged over all the path vertices in the aggregate. We assume that the emitted radiance \( L_e(y, \bar{y}) \) only depends on the position \( y \) on the emitter and not the path vertex \( x \), meaning \( L_e \) is constant over the aggregate. The remaining terms are the geometry term \( G \) and the scattering function \( \rho \), which we describe in more detail below. We illustrate the general idea of the aggregate target function in Fig. 9.4.

### 9.2.2.1 Geometry Term

The geometry term consists of the product of the inverse square distance and two dot products:
\[
G(x, y) = \frac{|N(x) \cdot \bar{x}| |N(y) \cdot \bar{y}|}{\|y - x\|^2}.
\]

We would like to compactly express the average of this term over all path vertices \( x \) in the aggregate in terms of closed-form functions that take \( y \) as parameter. We begin by approximating the concrete positions and normals of the vertices in the aggregate with simple geometric primitives: The ball \( B(\langle x \rangle, \langle r \rangle) \) and cone \( C(\langle N \rangle, \langle \theta_N \rangle) \) that represent the mean position \( \langle x \rangle \) and normal \( \langle N \rangle \) in the aggregate, as well as the average distance \( \langle r \rangle \) and angle \( \langle \theta_N \rangle \) from the mean:
\[
\langle x \rangle = \frac{1}{k} \sum_{i=1}^{k} x_i \quad \text{and} \quad \langle N \rangle = \frac{\sum_{i=1}^{k} N(x_i)}{\| \sum_{i=1}^{k} N(x_i) \|},
\]
\[
\langle r \rangle = \frac{1}{k} \sum_{i=1}^{k} \| x_i - \langle x \rangle \| \quad \text{and} \quad \langle \theta_N \rangle = \frac{1}{k} \sum_{i=1}^{k} \arccos N(x_i) \cdot \langle N \rangle,
\]

where \( x_1, \ldots, x_k \) are all the path vertices in the aggregate.

The geometries mentioned above depend exclusively on the path vertices and can be precomputed. However, we additionally require a representation of the distribution of directions \( \bar{x}_i \bar{y} \), which depend both on the path vertices and the sample \( y \). We
approximate this distribution with the cone $C(\langle \psi \rangle, \langle \theta_D \rangle)$, where $\langle \psi \rangle = \langle x \rangle y$ is the mean direction from $y$ to points on the sphere, and $\langle \theta_D \rangle$ is the cone angle subtended by the sphere at $y$:

$$\langle \theta_D \rangle = \arcsin \left( \frac{\langle r \rangle}{\| \langle x \rangle - y \|} \right)$$  \hspace{1cm} (9.14)

This cone can be computed cheaply during evaluations of the target function, and does not require access to the path vertices.

Assuming this geometry, we can compute the average squared distance of points in the ball from a given sample $y$:

$$\langle d^2 \rangle(y) = \frac{1}{|B(\langle x \rangle, \langle r \rangle)|} \int_{B(\langle x \rangle, \langle r \rangle)} \| x - y \|^2 \, dx = \frac{3}{5} \langle r \rangle^2 + \| \langle x \rangle - y \|^2. \hspace{1cm} (9.15)$$

We take a slightly different approach for the foreshortening term, as computing the average of an absolute dot product can be inconvenient. We note that both the mean value and the maximum value are valid choices for aggregating terms, and the latter leads to a compact closed-form solution. Given a direction $\omega$ and a cone defined by mean direction $\psi$ and cone angle $\theta$, the maximum value attained by the dot product over the cone is

$$\langle \cos \rangle(\omega, \psi, \theta) = \max_{\omega' \in C(\psi, \theta)} \omega \cdot \omega' = \cos(\max(\arccos(\omega \cdot \psi) - \theta, 0)). \hspace{1cm} (9.16)$$

Given these building blocks, we can now assemble the aggregate geometry term, swept over (approximations of) all path vertices in the aggregate:

$$\langle G \rangle(y) = \frac{\langle \cos \rangle(\langle N \rangle, \langle \psi \rangle, \langle \theta_N \rangle + \langle \theta_D \rangle) \cdot \langle \cos \rangle(\langle N(y) \rangle, \langle \psi \rangle, \langle \theta_D \rangle)}{\langle d^2 \rangle(y)}. \hspace{1cm} (9.17)$$

The first aggregate foreshortening term in the numerator computes the maximum dot product, swept over both the cone of normals at the path vertices, and the cone
of directions from path vertices to the sample. For the second foreshortening term, the normal $N(y)$ at the sample is unique, and only the cone of directions between $y$ and the path vertices is considered.

### 9.2.2.2 Scattering Function

In the general case, computing faithful aggregate representations of scattering functions is a challenging problem. However, because we only use the aggregate representation to drive sampling, we do not need it to be visually indistinguishable from the ground truth, and approximations are acceptable. We choose to represent the aggregate scattering distribution as a linear blending of a diffuse (view-independent) component and a view-dependent microfacet distribution $[139]$, with linear blend coefficient $\beta$ (where $\beta = 0$ means purely diffuse scattering).

The diffuse component is parametrized with albedo $\alpha$, and we choose the SGGX distribution $[61]$ to represent the microfacet component, which is parametrized with a symmetric matrix $S$. The average of multiple SGGX distributions can be closely approximated as the average of their coefficient matrices. This leads us to our aggregate scattering function

$$\langle \rho(y) \rangle = \langle \alpha \rangle (1 - \langle \beta \rangle) + \langle \beta \rangle \text{SGGX}((S)),$$

where $\langle \cdot \rangle$ stands for the straightforward average of the parameters over the path vertices.

If the renderer uses the same material model as the target distribution (i.e. a combination of a diffuse and microfacet distribution), then the parameters of our aggregate model can be readily computed. Otherwise, materials first need to be projected into this representation (see $[61]$) before the corresponding parameters ($\alpha, \beta, S$) can be averaged.
9.2.2.3 Combined Aggregate Target Distribution

Putting all of the building blocks of this section together, we obtain the aggregate target distribution

$$\hat{\rho}(y) = L_c(y) \langle G \rangle(y) \langle \rho \rangle(y).$$  \hspace{1cm} (9.19)

This distribution can be efficiently evaluated, stored compactly—needing only a few vectors and floats to describe average positions/directions etc.—and is expressive enough to represent useful target distributions for large aggregations of path vertices.

9.2.3 Reservoir Reuse with Aggregate Reservoirs

Given its target distribution, we now turn to the problem of performing resampling and reuse with an aggregate reservoir. In many ways, performing reuse with an aggregate reservoir is very similar to reuse in ReSTIR (Chapter 8): Aggregate reservoirs can be combined with reservoirs from previous frames or nearby reservoirs in the same frame using e.g. our pairwise MIS technique (Section 9.1.3), and we describe details later in this section.

However, aggregate reservoirs suffer from a bootstrapping problem that must be dealt with carefully. Because the target distribution of the aggregate reservoir only approximates the mean or maximum of the target distributions of the path vertices in the aggregate, it may be non-zero in parts of the domain even where all input target distributions are zero. This is problematic, because it means that for parts of the domain of the aggregate reservoir, none of the input reservoirs will generate any samples. Even with the bias removal strategies we introduced, this would lead to bias in subsequent reuse operations, because there is no canonical technique that is guaranteed to cover the entire domain.
To solve this problem, we kickstart the aggregate reservoir with a set of samples that are guaranteed to sample the entirety of the aggregate target distribution. We do this by generating an auxiliary sample $z$ at each path vertex that is distributed uniformly over the area of all emitters. The auxiliary samples are stored in the spatial data structure alongside the initial reservoirs. When the aggregate reservoir is populated, we initialize it by first resampling all auxiliary samples; the output sample is then guaranteed to cover the whole domain, and becomes the canonical technique in all subsequent resampling steps. After the initialization, we can perform reservoir reuse as normal and maintain unbiasedness.

**Temporal Reuse.** To perform temporal reuse, we use the center of the aggregate (i.e. $\langle x \rangle$ from Section 9.2.2) to look up into the spatial data structure from the previous frame and retrieve the aggregate reservoir at the same position. The aggregate reservoir from the previous frame is then combined with the current frame’s aggregate reservoir using pairwise resampling. Because the reservoirs are anchored in scene-space, there is no need for backprojection to compensate for camera motion. It would be possible to use the average motion of the geometry inside the aggregate to offset lookups in order to compensate for object motion, but it is not immediately clear if this would be beneficial; for example, it may be that the object is moving but the light sources are not, and motion compensation may be harmful instead. For simplicity, we do not do any motion compensation.

**Spatial Reuse.** To perform spatial reuse, we sample random locations distributed around the center of the aggregate, obtain the reservoirs there from the spatial data structure and combine them with the aggregate reservoir using pairwise resampling. Because reservoirs reside primarily near surfaces, we find it beneficial to bias the distribution of these locations to align with the geometry observed by the reservoir. We do this by first sampling a random location in the ball $\mathcal{B}(\langle x \rangle, \langle r \rangle)$ and a random
normal from the cone $C(\langle N \rangle, \langle \theta_N \rangle)$ (Section 9.2.2), and center a disk at the sampled location orthogonal to the sampled normal. We then uniformly sample a location on this disk and use this to query the spatial data structure. If the geometry observed by the aggregate lie primarily on a plane, then query locations will also primarily follow that plane; on the other hand, less structured geometry such as foliage will lead to samples distributed isotropically around the center of the aggregate. We choose the radius of the disk to be 4 times the radius of the ball in order to adapt to the scale of the scene.

### 9.2.4 Choice of Data Structure

The primary goal of the data structure in our algorithm is to subdivide space into a non-overlapping set of buckets that store aggregate reservoirs and their input data. While the choice of data structure does not influence the variance of the algorithm directly, it is crucial to its efficiency, and we will detail our choice of acceleration structure and the reasoning behind it in this section.

Spatial data structures have a rich history both in- and outside of graphics, and we have a large selection of potential algorithms to choose from. Because we aim to support dynamic scenes and we target graphics hardware for our algorithm, we must be able to rebuild or maintain the data structure at the start of each frame on the GPU. Additionally, we would like the data structure to automatically adapt to the local density of path vertices, so that each bucket contains roughly the same number of vertices. This has the benefit of concentrating resolution where it is needed, while also balancing the work loads across buckets and allowing our algorithm to adapt to the scene scale without requiring extra parameters.

One potential choice are hash-based algorithms, where locations in space are turned into keys that map into a hash table. While these have been used successfully [8] in light transport simulation, their implementation can become complex,
especially when parallel construction and spatial adaptivity is required. Instead, we are inspired by the work of Laine et al. [89] and choose an octree-based data structure. In order to enable fast parallel construction, we choose a dense octree for its simplicity.

### 9.2.4.1 Tree Structure Overview

Because we make use of a dense octree, each node in the tree is either a leaf node or has exactly 8 children. The spatial extents of each node are represented by an axis-aligned bounding box, where the root node covers exactly the bounding box of the scene, and a parent node is subdivided into 8 equally sized child nodes. The lack of adaptivity in the spatial extents of each node allows for a compact and simple data structure, and we conform to the distribution of path vertices purely through the structure of the tree, i.e. by deciding whether a node should be a leaf node or should be further subdivided.

Each leaf node in the tree stores a counter that determines how many path vertices are located in this leaf. During construction, we traverse the tree for each path vertex in the scene and increment the counter of the leaf node that contains the vertex. If the counter exceeds a threshold, the leaf is subdivided into eight children. While this heuristic is simplistic, it allows parallel construction of the tree (with some care) and maintains the desired property that leaf nodes contain roughly the same number of path vertices.

### Data layout

The data of our octree is stored in two arrays: The first array stores the nodes of the tree, whereas the second array stores the aggregate reservoirs associated with each leaf node. The use of arrays allows references to nodes and reservoirs to be represented by simple array indices. As we will see below, the maximum size of
the octree is both predetermined and manageable, and we preallocate the arrays to accommodate the largest representable octree to avoid resizing of the arrays at runtime.

Our choice of a simple data structure was deliberate so that each node in the tree can be represented with a single 32 bit integer (see figure below), which we call the node descriptor. Because the node descriptor maps to a machine integer, it can be manipulated with atomic instructions, allowing for lock-free parallel tree maintenance.

The structure of the tree is determined by the leaf bit, which differentiates interior nodes from leaf nodes. If the node is an interior node,\texttt{child\_idx} represents the array index at which the first child node is stored. All child nodes are stored consecutively, so that all children can be addressed with \texttt{child\_idx} + i with i < 8. If the node is a leaf node, \texttt{child\_idx} represents the index of the aggregate reservoir data structure associated with the leaf.

Because we allocate 23 bits to \texttt{child\_idx}, we can accommodate octrees with up to 8M nodes. The counter of the number of path vertices in a leaf nodes is allocated 7 bits, which allows us to account for up to 1B path vertices per frame. This is more than sufficient for real-time purposes on current hardware, but may need revision as hardware capabilities evolve.

\textbf{9.2.4.2 Tree Maintenance}

Because the octree structure remains largely the same between frames, we initialize the octree of the current frame with a copy of the previous frame’s octree with all leaf counters set to zero. We then maintain this data structure in several sequential passes, which we detail below.
Expansion Pass. In this pass, we iterate over all path vertices in the scene in parallel, search the tree for the leaf node that contains the vertex, and atomically increment the leaf node counter. If after incrementing the counter is equal to the subdivision threshold, the thread responsible for the last increment subdivides the node by allocating 8 children, constructing a new descriptor describing the subdivided node and atomically swapping it with the node’s previous descriptor. This ensures that threads currently traversing the octree always see a consistent data structure.

While a node is being subdivided, other threads may continue to arrive and increment the leaf counter. Because the leaf counter is only allocated 7 bits, it can wrap around and subsequent threads may believe they are responsible for subdividing the node. An additional lock bit is allocated in the node descriptor and atomically set when a node is being subdivided, so that it cannot be subdivided twice.

Pruning Pass. We copy the octree data structure from frame to frame mainly for performance reasons; if we started with an empty octree each frame, there would be significant contention in nodes close to the root in the expansion pass. However, after subdivision, the octree must be pruned to remove leaf nodes copied from the previous frame that did not receive enough path vertices this frame. To achieve this, we iterate over all nodes in the octree in parallel whose children are exclusively leaves, and merge the children into the parent if the sum of their counters is below some threshold. We choose the merging threshold to be half the subdivision threshold to give the tree hysteresis and keep the structure stable across frames.

9.2.5 Implementation Details and Results

We implemented our proposed algorithm in Falcor [6], the same rendering framework used for ReSTIR in the previous chapter, and test our algorithm on the same scenes as the previous chapter. All results are obtained on a machine with an AMD
Ryzen 3900X 12-Core processor and an NVIDIA RTX 3090 graphics card. The average time spent maintaining the octree in each frame amounts to 0.3 ms, and is not significant compared to the cost of the other parts of the algorithm. Although our algorithm supports estimating direct lighting at vertices outside the camera view, our implementation does not currently support constructing such indirect paths, and we restrict our evaluation to the same paths (i.e. length 2) supported by ReSTIR.

9.2.5.1 Implementation Details

We choose a subdivision threshold of 64, so that each bucket in our data structure contains at most 64 path vertices. This threshold was chosen as a trade-off between performance and variance: Aggregating too few vertices leads to a deep octree and many aggregate reservoirs and corresponding increase in render time, whereas aggregating too many vertices makes aggregate reservoirs less useful, increasing variance.

Because we aggregate over many vertices, the number of initial candidates produced at each vertex does not need to be very high. We sample $M = 2$ candidates for the initial reservoir at each path vertex; we did not observe improvements in variance for larger values of $M$. We found it useful to store multiple reservoirs in each spatial bucket to represent a more diverse sample pool; similar to ReSTIR from the previous chapter which stored $N = 4$ reservoirs at each pixel, we store $N = 8$ reservoirs in each bucket and process them in parallel.

9.2.5.2 Results

Before we evaluate our algorithm, it is useful to discuss how we expect its performance to compare to ReSTIR. Because our proposed algorithm operates in scene-space, we would hope it to show improved performance on complex geometry such as foliage, where screen-space algorithms struggle due to aliasing. At the same time, the effective resolution of reservoirs in our proposed algorithm is much lower: Up
to 64 pixels worth of reservoirs are aggregated into a single aggregate reservoir, and resampling occurs on target functions aggregated over a significant spatial footprint. Because of this, both variance and bias may increase for high-frequency effects such as sharp shadows or glossy reflections.

In Fig. 9.5, we show a false color visualization of our octree data structure in the Bistro scene. We assign each leaf node in the octree a random color and show the resulting images from two different perspectives. Fig. 9.5 (left) shows the distribution of nodes from the perspective of the camera used to trace the paths in this scene. The nodes all appear roughly equal-sized, each representing aggregates of approximately 64 path vertices. Fig. 9.5 (right) shows the same octree from a birds eye perspective, demonstrating how the resolution of the octree automatically adapts to the density of path vertices in the scene: Points close to the camera exhibit many small nodes, whereas nodes far from the camera naturally aggregate over larger spatial areas. Notably, this happens despite the octree having no internal knowledge of the camera; nodes simply adapt to the density of path vertices.

In Fig. 9.6, we compare equal-time renderings and RMAE graphs of our proposed octree-based ReSTIR to a modified (biased) ReSTIR baseline that uses both pairwise MIS and BSDF sampling (Section 9.1.3). We present two versions of our algorithm:
Figure 9.6: Top: Comparison of roughly equal time renderings of our octree-based ReSTIR to a modified version of ReSTIR that uses pairwise MIS and BSDF sampling as a baseline. Bottom: Evolution of RMAE over render time for the same algorithms. We show two versions of our scene-space algorithm: The ST-Octree, which performs spatiotemporal reuse; and the T-Octree, which only performs temporal reuse. The ST-Octree has significantly lower variance, but the rapid propagation of bias causes its RMAE to stagnate at longer render times. The T-Octree keeps bias constant across frames and shows similar performance to ReSTIR in the BISTRO scene, but is significantly noisier in the SUBWAY scene.
The ST-Octree, which performs spatiotemporal reuse; and the T-Octree, which only performs temporal reuse. As expected, the ST-Octree performs better in areas with complex geometric details, such as the foliage in the Bistro scene, or the grate in the Subway scene.

Surprisingly however, it also outperforms ReSTIR in terms of RMAE at render times below 20-30 ms. It appears that aggregation has a significant variance-reducing effect. Simultaneously, the number of aggregate reservoirs is much lower than the number of pixels, which reduces the computational cost of reuse operations. However, because each aggregate reservoir has significant spatial extent, the bias introduced from visibility reuse spreads quickly across frames, and at longer render times the bias causes the RMAE to stagnate.

For comparison, we also show results from a T-Octree that only performs temporal reuse and limits the spread of bias to within an aggregate reservoir. While this algorithm exhibits significantly more variance, the slope of its RMAE appears identical to that of ReSTIR, which suggests that spatial reuse, combined with the spatial extent of the reservoirs, causes the significantly increased bias.

9.3 Discussion, Analysis and Future Work

We conclude this chapter first with a short summary of our contributions and immediate limitations of our algorithm. We then close by pointing out two interesting issues present in ReSTIR and the scene-space extension of ReSTIR. We will spend some time analyzing these issues in Section 9.3.3 and Section 9.3.4 and suggesting potential solutions. These solutions are not currently practical or competitive, but form fruitful avenues for future work.
9.3.1 Contributions

In this chapter, we have presented the following contributions:

- We introduced a new derivation of RIS from a secondary MC estimator, and showed how different forms of RIS can be obtained by starting with estimators.

- Using this flexibility, we introduced a new MIS algorithm that we term pairwise MIS that provides significant benefits over the MIS strategy from Chapter 8, both in terms of variance and computational cost.

- We then introduced a new scene-space extension of ReSTIR. This algorithm operates on aggregate reservoirs anchored at scene locations that represent multiple path vertices using a practical aggregate target distribution. Compared on direct lighting, our scene-space extension shows benefits at real-time rates compared to ReSTIR, even though it uses significantly fewer reservoirs.

Both pairwise MIS and our scene-space extension unlock many potential avenues for future work, such as extensions to indirect lighting or resampling of challenging integrands. We discuss two possible extensions at the end of this section.

9.3.2 Limitations

There are a number of limitations to the algorithms we presented in this chapter.

Pairwise MIS relies on the assumption that a canonical sampling technique exists and provides reasonable sampling coverage over the entire domain. There are many conceivable scenarios where either no canonical technique exists, or is itself responsible for high variance. In Section 9.2, we already encountered a scenario where no canonical sampling technique was immediately available for the aggregate reservoir, and the reservoir needed to be bootstrapped by additional uniform samples to form
the canonical sample. Similarly, if all available techniques exhibit high variance, it may be better to use the standard balance heuristic for robustness, rather than use pairwise MIS.

Although our scene-space extension to ReSTIR shows improved variance and lower RMAE in the scenes we tested, it is not a universal improvement over ReSTIR. Because of the spatial aggregation of reservoirs, high-frequency effects such as near-specular materials would be poorly handled by this variant of ReSTIR, and the original algorithm may be preferrable. In addition, the added bias of the scene-space variant may be objectionable. It may be possible to bridge this gap by adding additional heuristics to the adaptivity criterion of our data structure. If the glossiness of the target distribution or the visibility complexity are taken into account during octree subdivision, the data structure could allocate higher resolution to challenging cases and, in the limit, reduce to behave identically to ReSTIR with a reservoir per pixel. Alternatively, we could increase the dimensionality of the octree to incorporate information about the scene geometry (e.g. surface normals). This would separate reservoirs with highly dissimilar geometry and potentially improve the quality of the target function, at the cost of fewer effective samples being contained in each octree cell.

9.3.3 Hierarchical Spatial Reuse

A perhaps unexpected challenge we encounter in the scene-space extension to ReSTIR from Section 9.2 that is not present in the original algorithm lies in spatial reuse. An advantage of operating in screen-space is that the image plane is covered densely in reservoirs, and it is easy for a pixel to discover a large number of reservoirs in its neighborhood. In contrast, the added dimensionality of scene-space means it is covered only sparsely in reservoirs: Reservoirs cluster on surfaces, and the bounding volume of scenes we consider in real-time is mostly empty. Even when neighboring
Figure 9.7: We illustrate various forms of spatial reuse in terms of graphs in 1D, where samples flow from left to right. The input reservoirs of the frame are shown on the left, where each circle is a reservoir and each edge represents reuse of the reservoir. In randomized reuse (a), each reservoir picks a few random reservoirs in the neighborhood and performs reuse. Here we show two repeated applications of the randomized spatial filter. In dense reuse (b), each reservoir resamples all reservoirs within a spatial neighborhood. Hierarchical reuse (c) does the same, but does so by iteratively downsampling and then upsampling the resolution of intermediate reservoirs to reduce the asymptotic cost. Linear hierarchical reuse (d) uses the same idea, but resamples with the two nearest lower resolution reservoirs in the upsampling stage to blur correlations across neighborhoods. Interleaved hierarchical reuse (e) interleaves multiple independent reuse trees to spread correlations over larger spatial extents and make them less visually apparent.

points are generated using the anisotropic distribution of Section 9.2.3, in practice we find that only 30–40% of queried locations contain any useful reservoirs.

An alternative to randomly sampling a small number of neighboring locations would be to deterministically search e.g. a dense grid of neighboring locations. While this yields a much larger number of neighboring reservoirs, it is prohibitively expensive due to the number of queries to the data structure. However, we can view the randomized and dense reuse as two examples in a much larger class of reuse patterns that we illustrate in Fig. 9.7(a,b) that inspire new forms of spatial filters.

**Reuse Hierarchies.** To solve this problem, we take inspiration from image filtering techniques, where filter convolutions with a large footprint can be achieved through repeated applications of smaller filters on progressively downsampled versions of the image. A similarly dense coverage in spatial reuse can be obtained by resampling in hierarchies: Applying the same idea to screen-space ReSTIR, we could iteratively perform resampling between adjacent neighbors followed by downsampling
the image to obtain progressively lower-resolution intermediate reservoirs that “see” increasingly larger neighborhoods. To propagate these samples back to the original (i.e. full-resolution) reservoirs, we follow the downsampling with resampling in the reverse direction (“upsampling”), from the intermediate lower-resolution reservoirs back to higher-resolution reservoirs (Fig. 9.7, (c)). The effective number of neighbors seen by each reservoir is much larger than in randomized reuse, but the intermediate stages of the algorithm operate on a much smaller number of reservoirs, which
provides considerable cost savings over naive dense reuse. Similar to the scene-space version of ReSTIR we introduced in Section 9.2, each intermediate node represents an aggregate reservoir, and it can be populated using the strategies we show in Section 9.2.2 and Section 9.2.3.

This algorithm suffers from major deficiencies that make it unappealing in practice. These stem from the fact that large neighborhoods of reservoirs receive samples from the same intermediate reservoirs (Fig. 9.7, (c)) during the upsampling process, causing significant correlation in the samples used for rendering. On the one hand, this manifests as square-shaped artifacts (Fig. 9.8 (b)) in the image: Within square neighborhoods, reservoirs receive samples from the same intermediate reservoirs and are correlated, and noise discontinuities appear between such neighborhoods.

**Linear Hierarchies.** We again take inspiration from image processing, and view the naive hierarchical reuse as a “nearest” upsampling filter, which only receives samples from one lower-resolution reservoir. If we instead use the resampling analog of linear filtering, where a reservoir receives samples from the four (in 2D) or eight (in 3D) closest reservoirs from the lower-resolution stage (Fig. 9.7, (d)), we can “blur” correlations across spatial neighborhoods, which reduces visible correlation artifacts somewhat (Fig. 9.8 (c)). However, there still exists visible spatial correlation in the output samples, which causes distracting “popping” artifacts in animation.

**Interleaved Hierarchies.** Our proposed solution to the correlation issue is to explicitly spread correlations over larger neighborhoods so they can no longer be perceived by human vision. We do this by assigning a class label to each input reservoir, and partitioning the input reservoirs into a separate hierarchy for each class label. Each hierarchy is independent, but all hierarchies are interleaved (Fig. 9.7, (e)), removing visually apparent correlation. We conceptually visualize this in a 2D example in Fig. 9.9, where we assign class labels between 1 and 4 to each pixel. Our suggested
solution is then equivalent to rearranging the input image into four separate images in which all pixels with the same class label are adjacent. We then perform (linear) hierarchical reuse on each image separately, and re-interleave the images to retrieve the ordering of the input image. In practice, we can obtain the same effect \textit{in situ} by remapping pixel coordinates, without having to create separate images.

When choosing class labels, we must trade off increase in bias and variance with decrease in correlation artifacts: For the least amount of correlation, we want as many class labels spread as far apart from each other as possible, whereas we want pixels of the same class to be in close proximity in the input to limit the spread of bias and reduce variance in the aggregate reservoir. We use the strategy of Boulos et al. [18] and distribute 64 class labels based on 8x8 pixel Sudoku tiles (Fig. 9.9). This yields pleasing noise characteristics and represented the best tradeoff between variance and correlations in our experiments.

\section*{9.3.3.1 Results and Discussion}

We implemented a 2D prototype of hierarchical reuse on top of the ReSTIR algorithm and compare both equal time renderings and RMAE graphs of randomized spatial-
only reuse to our hierarchical spatial reuse in Fig. 9.10. At first glance, hierarchical reuse provides significant benefit in terms of visual variance and RMAE reduction. This is especially so in challenging areas with complex geometric features, where the much denser coverage of the input provided by hierarchical reuse yields more useful reservoirs for reuse than randomized reuse.

However, the larger spatial footprint required by hierarchical reuse to spread correlation effects, and the infeasibility of visibility debiasing due to the use of aggregate
reservoirs leads to hierarchical reuse suffering from significantly increased bias compared to the randomized reuse of ReSTIR. When combining the respective spatial reuse algorithms with temporal reuse, hierarchical reuse leads to intolerable spread of bias and no longer provides any benefit over randomized reuse in terms of RMAE.

Furthermore, we did not find hierarchical reuse to provide any benefit in the scene-space version of ReSTIR, even though this use case initially inspired the use of hierarchies. The reason for this is that aggregate reservoirs in scene-space already have a significant spatial footprint. Using hierarchical reuse while avoiding correlation artifacts in this scenario would lead to spatial reuse over very large spatial distances, and the increase in both bias and variance do not outweigh the benefit of hierarchical reuse.

Because of this, hierarchical reuse is not yet a practical improvement to either screen-space ReSTIR or scene-space ReSTIR. It may be possible that improved class label distributions (using e.g. the work of Jarosz et al. [68]) or alternative hierarchy structures may reduce correlations enough that smaller neighborhoods in the input can be used, removing the disadvantage of hierarchical reuse.

9.3.4 Reservoir Overuse

We close this chapter with a brief analysis of the reuse patterns that arise in spatiotemporal reservoir resampling, and how they can lead to overrepresentation, or overuse, of certain reservoirs.

In Fig. 9.11 (a), we show a simplified version of ReSTIR in a 1D domain. This allows us to visualize both space (individual frames) and time (present and past frames) in a single image: The first frame of the animation is shown on the left, and the current (most recent) frame on the right. Each frame corresponds to a vertical column of “pixels”. We modify ReSTIR to use only a single reuse pass, which combines both spatial and temporal stages by resampling the three closest reservoirs from the
previous frame. We analyze a single reservoir from the current frame (shown in red), and visualize all reservoirs from all past frames that contribute to that reservoir as well as the paths along which samples flow across frames to the current frame. For illustrative purposes, we set the target function to a constant 1, so that reservoir reuse in this scenario reduces to simply choosing randomly between the input reservoirs based on their respective $M$ (using the weighting scheme of Chapter 8).

In Fig. 9.11 (b) and (c), we simulate 10'000 runs of ReSTIR in this setup and record a histogram that shows which reservoir originally generated the sample that is stored as the output sample in the red reservoir in the current frame. In Fig. 9.11 (b), we allow the $M$ in each reservoir to grow without limitation; in Fig. 9.11 (c), we clamp each reservoir’s $M$ after reuse to a maximum of 20 times the number of initial candidates at the reservoir (identical to Chapter 8).

The ideal histogram we would expect is a uniform distribution over all yellow nodes from Fig. 9.11 (a). However, surprisingly, in either scenario samples from the first few frames dominate the samples in the current frame. If we allow $M$ to grow unboundedly, samples come exclusively from the first few frames; if we clamp $M$, samples from more recent frames are also sometimes represented in the output, but are still in the minority.
We can explain this unexpected behavior as follows: Weighting a reservoir’s sample by the reservoir’s effective \( M \) would be the correct choice if the reuse kernels of adjacent pixels did not overlap; in this case, the corresponding reuse graph in Fig. 9.11 (a) would be a tree, and there would only be a single path connecting each reservoir to the reservoir in the current frame. However, because the kernels do overlap, we can see from Fig. 9.11 (a) that there are in fact many paths that connect a reservoir from early frames to the current frame. Each path represents one “chance” of that reservoir’s sample being chosen as the output sample. In this simplified 1D scenario, the number of paths grows exponentially with time, which explains why samples from early frames dominate the output. Clamping \( M \) puts a bound on the weight of early samples, but it only partially alleviates the problem. Aggressive clamping would be needed to solve the problem, which at the same time would effectively disable temporal reuse.

This is not merely a theoretical problem, but can be observed in practice as well. In Fig. 9.12, we show a rendering of the Subway scene that was modified to be particularly challenging to render. We changed each triangle in this scene to be emissive to simulate indirect lighting, leading there to be 3.6M emitters. Simultaneously, we lowered the number of initial candidates to 1 and disabled BSDF sampling. As we would expect, the output of ReSTIR (Fig. 9.12 (b)) exhibits high variance in this case, but the mode of failure is particularly interesting. In Fig. 9.12 (c), we show a false coloring of ReSTIR, where each pixel is colored based on the sample it uses for shading. We would expect uniform color noise over the whole image, but instead we can observe distinct clusters of pixels that all use exactly the same light sample. In all cases, the origin of the sample is a frame far in the past, which dominates the output of the current frame due to the mechanism we explained earlier.

Although this is a contrived example, we can observe this example in any failure case of ReSTIR: Where not enough good samples are available, sample diversity is severely diminished and lone samples can dominate the output of large pixel
neighborhoods, leading to visible correlation artifacts and flickering across frames. This is caused by an interaction of $M$ weighting and overlapping reuse kernels.

It is not immediately clear how to fix this behavior in ReSTIR. There are several potentially fruitful avenues; special non-overlapping reuse kernels, similar to the interleaved trees of Section 9.3.3, could prevent overrepresentation of old samples, or different weighting/clamping strategies of $M$ that account for kernel overlap could do the same. We have not found straightforward applications of these ideas to work,
and we believe ReSTIR could be considerably improved if this problem was resolved, making it an appealing target for future work.
Part III

CONCLUSION

So much universe, and so little time.

— Terry Pratchett
CONCLUSION AND OUTLOOK

In this dissertation, we sought to improve the accuracy of physically based rendering both by modifying the model of light transport, as well as the algorithms used to solve it. We achieved this goal by means of adding and exploiting correlations in light transport problems. We began in Chapter 4 by noting that existing models of transport in participating media ignore the natural correlations between the location of scatterers in many media occurring in nature. We showed that these correlations demonstrably modify the transmittance in such media to no longer conform to the traditional exponential model, and we could show that incorporating such non-exponential transmittances into the classical model of radiative transfer is non-trivial. We were able to derive a new transport model as a series of approximations to an ensemble average of stochastic and non-stationary heterogeneous media. The core of the resulting model is formed by four interrelated functions that describe collisions in media. The model is simple and practical, and existing renderers can be easily augmented to also incorporate correlated scatterers.

We then turned our focus to simulating transport in participating media. We looked to photon mapping based estimators, which reuse large numbers of light paths. We showed how prior beam estimators can be viewed as replacing distance sampling steps (“collision” estimators) with their expected values (“expected value” and “track-length” estimators), and can be generalized to plane and parallelepiped estimators by replacing additional distance sampling steps with their corresponding expected values. These estimators provided significant reduction in error over prior
work, both theoretically and empirically. Finally we showed how these estimators could be viewed as special cases of a reformulation of the path integral where three dimensions are integrated analytically, which unlocks additional classes of estimators.

Inspired by the efficiency gains provided by additional reuse in photon mapping, we considered perturbations in Primary Sample Space MLT. We showed that the existence of multiple sampling techniques in BDPT poses a challenge for techniques such as Multiplexed MLT that attempt to transition between different techniques in primary sample space. We introduced the framework of Reversible Jump MCMC to graphics, which allowed us to bridge path space and primary sample space. Based on this relation between the two spaces, we then introduced a new perturbation to Multiplexed MLT that can effortlessly transition between sampling techniques using reversible jumps between subspaces, and termed this new algorithm Reversible Jump MLT.

The ease of transitions in RJMLT provided significant efficiency gains due to the additional reuse permitted by such an algorithm. The cost of reuse, however, is the presence of visually correlated noise in the image. Where the complexity of light transport—or lack thereof—does not warrant the drawbacks of visual correlations, the additional machinery provided by algorithms like RJMLT provides little benefit. We pointed out that transport in many scenes consists of a mixture of both difficult and easy transport, and considered the benefit of separating the two to be handled by algorithms of different sophistication. We achieved such a separation with a new selectively metropolized estimator, which reformulates MIS in primary sample space as simple firefly detectors. Given such a detector, we presented a new algorithm that selectively handles simple transport using an efficient algorithm like path tracing, and only passes difficult transport to an MLT based estimator.

Motivated by the introduction of ray-tracing capable graphics hardware, we then turned to the problem of efficient reuse in real-time applications. We noticed that the
algorithms we introduced in prior chapters are not a good fit to this problem, as they require evaluation of the measurement contribution for each reused sample. Given the limited number of ray-tracing operations afforded by the hardware, this would severely limit the amount of reuse possible, and conversely limit efficiency. Based on resampled importance sampling, we introduced a new rendering algorithm based on the resampling and reuse of reservoirs across space and time. Given a few number of initial samples at each pixel drawn from a simple distribution, we incrementally improved the distribution of these samples to obtain low-variance direct lighting at real-time rates, while only tracing a few rays.

The resulting algorithm provided significant efficiency gains over state-of-the-art methods, and using these ideas as a base, we proceeded to propose several extensions and improvements. We rederived RIS to enable the inclusion of pairwise MIS weights that further reduce variance when combining dissimilar reservoirs. Using the pairwise mechanism, we further generalized reservoir reuse to operate on scene-space locations, rather than being bound to image space. This required the introduction of aggregate reservoirs, which combine the concerns of multiple sampling locations. We closed with an analysis of the residual problems of reservoir reuse, and proposed several extensions that provide fruitful directions for future work.

10.1 Outlook

The rendering algorithms we introduced in this dissertation all take advantage of computation reuse across pixels. These different forms of reuse are what allowed for increased efficiency, compared to naively estimating individual pixel values using Monte Carlo simulation. At the same time, the existence of these algorithms points to a much more fundamental property about our current models of light transport. The only reason efficiency gains through reuse are possible is because the representations
of radiance in the scene afforded by our current models are redundant. Individual radiance measurements across the image plane are assumed to be wholly separate, and similarly, the radiance distribution in the scene is allowed to be of arbitrary frequency and structure.

In practice, these assumptions are mostly untrue. All rendering algorithms based on reuse exploit this property, and in effect use local models of light transport to guess at the distribution of radiance. These local models may present differently, such as the proposal kernel in algorithms based on MCMC, or the shape of photons in algorithms based on photon mapping, but in a sense, all of these algorithms substitute lower-dimensional representations of light transport for the general radiance function $L(x, \omega)$; where the representation is accurate, efficiency is gained. Another compelling example of this is the considerable effectiveness of denoising algorithms, which project rendered estimates of radiance into lower-dimensional representations of the image, where representations are either predetermined or learned from data. These representations are not expressive enough to capture noise, and result in images with most of the undesirable variance removed.

Common to all these algorithms is a removal of excess degrees of freedom, so that renderings can only represent the desired signal (the rendered image) rather than undesired signals (such as noise). A very related problem exists in appearance models that represent the bulk behavior of microscopic detail assumed to be stochastic. We first saw an example of this in Chapter 4 for participating media, but e.g. microfacet models and related scattering distributions solve an identical problem. In both of these cases, the detail of the underlying geometry is assumed to be microscopic so that it cannot be resolved after measurement by pixels. The costly, detailed representation is then replaced with an affordable, low-dimensional representation (e.g. an exponential, or an analytic microfacet distribution) that eliminates the cost of the original model.
Although rendering algorithms and geometry representation initially appear to be orthogonal problems, they seem to solve interrelated problems and would benefit from a unified approach. In recent years, a growing body of research has been dedicated to aggregate representations for arbitrary (i.e. non-stochastic) geometry. Similar to the intent of models of participating media or microfacet surfaces, the goal is to filter redundant detail and only represent bulk behavior resolvable by the sensor model. Of increasing importance is aggregation of not only geometry, but also light transport, to capture interactions with matter not explicitly represented. The author believes that a unified representation of level-of-detail for geometry and light transport is an important future step to obtain affordable all-scale light transport. In an ideal world, all of the specialized reuse algorithms we presented in this dissertation would no longer be required, and the underlying model of light transport and geometry would not have the expressive power to express redundant information to begin with. We believe this to be an important and fruitful direction for future work.
Part IV

APPENDIX

“Are you afraid?”
“Yes.”
“Energy never stops, remember. It just changes forms.”
“I am still afraid.”
— Amie Kaufman
This appendix contains additional derivations to supplement derivations in the main text. Appendix A.1 is a derivation of our non-exponential transport framework that operates on discrete scatterers and does not rely on the classical RTE. In Appendix A.2, we derive a control variate estimator for low-noise transmittance estimation in blurred photon planes.

### A.1 Non-exponential transport in particle distributions

The derivation in Section 4.3 relies on the classical radiative transport theory to define the realizations of the random process over which the macroscopic problem is averaged. This may appear to limit the generality, but in this section we start from a weaker set of assumptions to show the same fundamental relationships between transmittance and free-flight PDF for light originating from the medium and from outside the medium still hold.

We consider a realization of the medium to be a binary volume in which transport is described simply by a visibility function $V_{\mu}(x, y)$ which is equal to 1 when the line segment from $x$ and $y$ is unoccluded and 0 when it is occluded. This allows for any arrangements of particles of any shape and orientation [63], with no limits on correlation between particles. Visibility is reciprocal: $V_{\mu}(x, y) = V_{\mu}(y, x)$; and it is monotonic: for a point $z$ on the line between $x$ and $y$, $V_{\mu}(x, y) = V_{\mu}(x, z) \land V_{\mu}(z, y)$; in particular, $V_{\mu}(x, y) \implies V_{\mu}(x, z)$ for all $z$ between $x$ and $y$. 
The expected value of $V_\mu$ over the ensemble of random volumes is the unconditional probability of visibility from one point to another, called the *simple transmittance* or just transmittance, and denoted FF:

$$FF(x, y) = \Pr(V_\mu(x, y) = 1) = E[V_\mu(x, y)].$$

FF inherits reciprocity and monotonicity from $V_\mu$: $FF(x, y) = FF(y, x)$ and $FF(x, z) \geq FF(x, y)$ for all $z$ between $x$ and $y$. In order to exclude some awkward cases we assume $FF(x, x) = 1$ and that FF is continuous. (This excludes infinitely thin opaque surfaces, which we consider separately in section 4.3, and also media that have a non-negligible packing rate.) Transmittance is appropriate for computing how much light makes it from a surface source to a surface detector through the medium: the surfaces are the same in all realizations.

To make later definitions simple we define a restriction of FF to a ray:

$$FF_{x, \omega}(s) = FF(x, x + s\omega) \quad (A.1)$$

The cross-section, or attenuation coefficient, is the probability per unit length of hitting the medium. In terms of $FF_{x, \omega}$, it is

$$\sigma(x, \omega) = \lim_{s \to 0} \frac{\Pr(-V_\mu(x, x + s\omega))}{s} \quad (A.2)$$

$$= \lim_{s \to 0} \frac{1 - FF_{x, \omega}(s)}{s} \quad (A.3)$$

$$= -FF'_{x, \omega}(0) \quad (A.4)$$

To simplify notation we write $FF'(x, \omega)$ for $FF'_{x, \omega}(0)$.

The free-flight distribution $f_\mu(x, y)$ is a probability density along rays that describes the distribution of the first intersection with the volume for light starting at $x$ and heading towards $y$. First intersecting at $y$ involves two events happening together: $y$
is visible from \( x \), and a point infinitesimally beyond \( y \) is not visible from \( x \). This lets us compute the probability, as a density along the line joining \( x \) and \( y \):

\[
fp(x, y) = \lim_{s \to 0} \frac{\Pr[V_\mu(x, y) \wedge \neg V_\mu(x, y + sx\hat{y})]}{s}
\]

(A.5)

\[
= \lim_{s \to 0} \frac{E[V_\mu(x, y)(1 - V_\mu(x, y + sx\hat{y}))]}{s}
\]

(A.6)

\[
= \lim_{s \to 0} \frac{E[V_\mu(x, y) - V_\mu(x, y + sx\hat{y})]}{s}
\]

(A.7)

\[
= \lim_{s \to 0} \frac{\text{FF}(x, y) - \text{FF}(x, y + sx\hat{y})}{s}
\]

(A.8)

\[
= -\text{FF}'(x, y)(|y - x|)
\]

(A.9)

This says that the probability per unit length of stopping at \( y \) is equal to the rate of decrease of transmittance along the ray from \( x \) towards \( y \). This derivative of \( \text{FF}_{x,\hat{y}}(s) \) with respect to \( s \) can also be seen as a directional partial derivative of \( \text{FF}(x, y) \) with respect to \( y \).

We define the notation

\[
\text{FF}'(x, y) = \text{FF}'_{x,\hat{y}}(|x - y|)
\]

to simplify writing these derivatives along the line joining two segments. Note that even though \( \text{FF}(x, y) = \text{FF}(y, x) \), in general \( \text{FF}'(x, y) \neq \text{FF}'(y, x) \).

Light that is emitted by or scattered by the volume itself experiences a different transmittance and a different free flight distribution. The \textit{volume-source transmittance} \( \text{PF}(x, y) \) is the fraction of light emitted by the volume at \( x \) towards \( y \) that reaches \( y \). In a random medium the volume-source transmittance from \( x \) to \( y \) is the probability that \( y \) is visible from \( x \) \textit{conditioned on the presence of a particle at} \( x \). Similarly, the \textit{volume-source free flight distribution} \( \text{pp}(x, y) \) is the length distribution of first intersections with the volume, also conditioned on an intersection at \( x \).
Because of the assumption of low packing rate, we can’t condition literally on a particle at \( x \); rather we look for an intersection in an infinitesimal interval beyond \( x \), just as when defining \( \sigma \).

We can calculate PF using the probability of visibility from \( x \) to \( y \) conditioned on having an intersection in a short interval before \( x \), in the limit as the collision interval shrinks. Using the familiar rule for conditional probability \( \Pr(P|Q) = \frac{\Pr(P \land Q)}{\Pr(Q)} \),

\[
PF(x, y) = \lim_{s \to 0} \frac{\Pr(V_\mu(x, y) \land \neg V_\mu(x + sy\hat{x}, x))}{\Pr(\neg V_\mu(x + sy\hat{x}, x))}
\]

(A.10)

\[
PF(x, y) = \lim_{s \to 0} \frac{\Pr(V_\mu(x, y) \land \neg V_\mu(x + sy\hat{x}, y))}{\Pr(\neg V_\mu(x + sy\hat{x}, x))}
\]

(A.11)

\[
PF(x, y) = \lim_{s \to 0} \frac{\Pr(V_\mu(x, y) \land \neg V_\mu(x + sy\hat{x}, y))}{\Pr(\neg V_\mu(x + sy\hat{x}, x))}
\]

(A.12)

\[
PF(x, y) = \lim_{s \to 0} \frac{(FF(x, y) - FF(x + sy\hat{x}, y))}{\Pr(\neg V_\mu(x + sy\hat{x}, x))}
\]

(A.13)

\[
PF(x, y) = \lim_{s \to 0} \frac{(FF(y, x) - FF(y, x + sy\hat{x}))}{\sigma(x, y\hat{x})}
\]

(A.14)

\[
PF(x, y) = \frac{-FF'(y, x)}{\sigma(x, y\hat{x})}
\]

(A.15)

Since \( \sigma(x, y\hat{x}) \) is the limiting value of \( -FF'(y, x) \) as \( y \) approaches \( x \), this normalization makes PF into a proper transmittance that starts at 1 and decreases to 0 with distance.

Since they are both derivatives of FF, it is clear that \( fp \) and PF are related by the reciprocity relation:

\[
PF(x, y)\sigma(x, y\hat{x}) = -FF'(y, x) = fp(y, x)
\]
Finally, we can calculate $pp(x, y)$ from PF in the same way we derived $fp$ from FF: the derivative along the ray of transmittance gives the corresponding free-flight PDF:

$$pp(x, y) = \lim_{s \to 0} \frac{PF(x, y) - PF(x, y + sx\hat{y})}{s} \quad (A.16)$$

$$= - \frac{PF'_x(|y - x|)}{\sigma(x, y\hat{x})} \quad (A.17)$$

$$= \frac{FF''(x, y)}{\sigma(x, y\hat{x})} \quad (A.18)$$

Since PF is already the derivative of FF along the ray with respect to $x$, $pp$ is a mixed second derivative of FF with respect to both $x$ and $y$, denoted $FF''(x, y)$ on the last line above.

The conclusion of this section is that the relationships between FF, $fp$, PF, and $pp$, which were established in Section 4.3 using an ensemble average over random heterogeneous classical media, also hold in the more general case of arbitrary assemblies of correlated particles.

### A.2 Control Variates for Blurred Photon Planes

Estimating the contribution of a blurred photon plane involves the computation of an integral along the interval of overlap between the camera segment and the photon plane. In order to reduce variance introduced by this estimation, we apply control variates [47] in our implementation of the photon plane estimator for homogeneous media.

The concept of control variates is simple. Say we wanted to estimate the integral of some function $f(x)$ over a domain $\Theta$ using Monte Carlo integration:

$$\int_{\Theta} f(x) \, dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \quad (A.19)$$
Integrating \( f(x) \) analytically may not be feasible, but say we had access to some other function \( g(x) \) whose integral \( G(\Theta) \) over the domain is known. \( g(x) \) is then referred to as the control variate, and we can form a new estimator of the integral of \( f(x) \) with

\[
\int_{\Theta} f(x) \, dx \approx G(\Theta) + \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i) - g(x_i)}{p(x_i)}. \tag{A.20}
\]

If \( g(x) \) is correlated with the integrand \( f(x) \), then the Monte Carlo estimator in Eq. (A.20) has less variance compared to the straightforward MC estimator.

We repeat the contribution of a blurred photon plane:

\[
\mathcal{D}_{Q-B1D}^{l-1,k} = \int_{s_{k-}}^{s_{k+}} g(\tilde{t}_{l-1}) g(\tilde{t}_l) \left\{ \frac{K_1(x_l, y_k)}{J_{Q-B1D}^{l-1,l}} \rho_{\omega}^{l,k} \right\} g(s) \, ds. \tag{A.21}
\]

Evaluating this expression involves computing an integral over the segment of overlap \([s_{k-}, s_{k+}]\) between the camera ray and the photon plane.

Because the directionality of the blur is orthogonal to the photon plane, the distance throughputs \( g(\tilde{t}_{l-1}) \) and \( g(\tilde{t}_l) \) implicitly depend on the integration variable \( s \), which complicates analysis slightly. We note that the blur direction of these two terms is not of major practical concern, and as a first simplification we replace \( \tilde{t}_{l-1} \) and \( \tilde{t}_l \) by their values at the center of the interval of overlap. This changes the directionality of the blurring, so that these quantities are blurred along the camera segment instead of the plane normal.

This allows us to move the majority of terms out of the integrand to obtain

\[
\mathcal{D}_{Q-B1D}^{l-1,k} = C \int_{s_{k-}}^{s_{k+}} g(s) \, ds, \quad \text{with} \quad C = g(t_{l-1}) g(t_l) \left\{ \frac{K_1(x_l, y_k)}{J_{Q-B1D}^{l-1,l}} \rho_{\omega}^{l,k} \right\}. \tag{A.22}
\]

The distance throughput along \( s \) is composed of a visibility and transmittance term, \( g(s) = V(s) \text{Tr}(s) \), where \( \text{Tr}(s) = e^{-\sigma s} \) in homogeneous media. In the context of
control variates, \( g(s) \) becomes the integrand of interest, and the transmittance \( \text{Tr}(s) \)
becomes the control variate \( g(x) \). We expand the integrand and rearrange to obtain

\[
\mathcal{D}_{\text{Q-B}1}\mathcal{B} = C \int_{s_{k-}}^{s_{k+}} V(s) \text{Tr}(s) \, ds \\
= C \left( \int_{s_{k-}}^{s_{k+}} V(s) \text{Tr}(s) - \text{Tr}(s) + \text{Tr}(s) \, ds \right) \\
= C \left( \int_{s_{k-}}^{s_{k+}} \text{Tr}(s) \, ds + \int_{s_{k-}}^{s_{k+}} (V(s) - 1) \cdot \text{Tr}(s) \, ds \right) \\
= C \left( \frac{e^{-\sigma s_{k-}} - e^{-\sigma s_{k+}}}{\sigma} + \int_{s_{k-}}^{s_{k+}} (V(s) - 1) \cdot \text{Tr}(s) \, ds \right) \\
\approx C \left( \frac{e^{-\sigma s_{k-}} - e^{-\sigma s_{k+}}}{\sigma} + \frac{1}{N} \sum_{i=1}^{N} \frac{(V(s_i) - 1) \cdot \text{Tr}(s_i)}{p(s_i)} \right)
\]

We can interpret the first term in Eq. (A.27) as the analytic integral of the contribution of the photon plane assuming full visibility. The second term is then a correction factor that accounts for occlusion along the camera segment.

In our implementation of short–short planes, we only take a single sample of the estimator in Eq. (A.27). We observe that in our test scenes, the vast majority of visibility tests (more than 95% on average) are unoccluded, and the estimate assuming full visibility is a good variate for the integrand. Fig. A.1 demonstrates the variance reduction of control variates compared to naive Monte Carlo sampling of the blurred photon plane contribution.
A.2.1 Relation to Photon Beams

Our control variate estimator shares some similarities with prior work by Jarosz et al. [70] on photon beams. Density estimation with Point×Beam (3D blur) involves a similar integration problem, and the analytic integral of the control variate in Eq. (A.27) is in fact identical to Eq. (11) in the paper by Jarosz et al. Unlike Point×Beam however, evaluating the contribution of a photon plane involves an additional visibility term, which explains the need for an additional Monte Carlo estimator in Eq. (A.27).

Although our implementation only supports photon planes, a similar control variate approach could be applied to rendering with photon volumes. However, the integrand in the photon volume case involves three transmittance terms that depend on the integration variable, and deriving an analytic expression for the control variate requires slightly more work. This shares some similarity with prior work on Beam×Beam (2D blur) estimators [70], which require evaluating integrals of two transmittance terms.
BIBLIOGRAPHY


[41] Euclid. Optics. 300BC.


DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification. Where research was performed as a member of a research group, I have made a substantial contribution to the work included in the dissertation.

Hanover, New Hampshire, August 2021

______________________________
Benedikt Bitterli