OPTIMIZED NUMERICAL SOLVERS FOR CALCULATING RADIATIVE TRANSFER VIA A FEYNMAN PATH INTEGRAL FORMULATION

A Dissertation

by

BRENNEN RAY TAYLOR

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2023

Major Subject: Computer Science

ABSTRACT

The simulation of visible light propagation and interaction within virtual environments is particularly interesting in computer graphics. Volume rendering, a crucial technique, aims to simulate light transfer through scattering media accurately. While existing solutions provide reasonable results, they become computationally complex when multiple scattering events occur. This paper introduces a numerical solver based on the Feynman Path Integral designed to capture high orders of scattering events. While previous solvers of the FPI were computationally inefficient, we present novel numerical approaches for solving the FPI, which offer improved performance. The solvers are validated and applied to render volumetric environments, demonstrating their effectiveness. The proposed solutions hold promise for simulating radiative transfer in various disciplines and could serve as a benchmark for high-order scattering simulations.

DEDICATION

To my mother, Wendi Taylor, who I wish could have seen this.

ACKNOWLEDGMENTS

Many helped me along the way on this journey. I want to take a moment to thank them.

First, I would like to thank my advisor, Dr. John Keyser, for all his guidance and counsel throughout these years. You have inspired me since we first worked together during my undergraduate studies, and I am extremely grateful for the opportunity to have studied under you. I also thank my committee members, Dr. Scott Schaefer, Dr. Nima Kalantari, and Dr. Ergun Akleman, whose advice and wisdom proved invaluable throughout my studies. I would also like to thank Dr. Jerry Tessendorf, a remarkable collaborator and teacher throughout my studies and with whom I've forged a great friendship through the years.

Next, I would like to thank my tremendous family; my parents, Ray and Wendi Taylor, who gave me a wonderful childhood and always believed I could achieve anything I set my mind to; my incredible step-mom, Lorelai Taylor, who has carried on this tradition and has been a huge support through my graduate studies; my sister Bailey and her husband Daniel, with whom I've been to hell and back and who inspire me to be better every day; my Grandma Frankenberger (GF) and Grandma Taylor (GT), who, in addition to their support, have always made the best brunch companions on all my trips home; and my step-siblings Devin, Josh and Kara, who have been wonderful additions to the family! To you and all the numerous other unnamed family members who have had my back all these years, I love you so much!

Lastly, I would like to thank the rest of my friends; my lifelong friends Trevor and Paul, lab mates, the boys, and the vet crew, among many others, who have helped keep me semi-sane through my grad studies. See, I told you I would finish school one day! I also have to give a special shout-out to my friend Jory, who has been my mentor and one of my best friends for years, and who first introduced me to undergraduate research. Thank you for everything!

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors John Keyser, Scott Schaefer and Nima Kalantari of the Department of Computer Science and Professor Ergun Akleman of the Department of Visualization.

The work in Chapters IV was published in collaboration with Professor Jerry Tessendorf of Clemson University. Additionally, the initial weight derivation of the simplified weighting kernel was performed in collaboration with Dr. Tessendorf.

The student completed all other work for the dissertation independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and a dissertation research fellowship from That Foundation. Additionally, this study was supported by the Haglar Fellows program, of which both myself and Dr. Tessendorf participated.

NOMENCLATURE

FPI	Feynman Path Integral
FPI-RT	Feynman Path Integral Formulation of Radiative Transfer
FPI-SW	Feynman Path Integral Formulation with Simplified Weighting Kernel
$G(\mathbb{R})$	A single path integral, parameterized by boundary conditions $\mathbb{R}$.
$\mathbb{R}$	Boundary conditions of FPI
$\mathbb{P}$	A valid path
$\sigma_b(\vec{\boldsymbol{x}})$	A function mapping a world position to a volume scatter coefficient.
$\sigma_a(\vec{\boldsymbol{x}})$	A function mapping a world position to a volume absorption coefficient.
$\sigma_t(\vec{\boldsymbol{x}})$	A function mapping a world position to a volume transmissivity coefficient.
Ω	The full solid angle
$\mathbb{S}$	A single segment of a path $\mathbb{P}$.
M	The number of equal-length segments a path is subdivided into.
Δs	The length of a single segment $\mathbb{S}$.
S	Arclength
$\vec{\boldsymbol{x}}_S$	Starting Position of Path
$\hat{\boldsymbol{\omega}}_S$	Starting Direction of Path
$\vec{\boldsymbol{x}}_R$	Receiver Position of Light
$\hat{\boldsymbol{\omega}}_R$	Receiver Direction of Light
$\boldsymbol{r}(s)$	A function mapping arclength to a world position, used to define a $\mathbb{P}$.
$\hat{\boldsymbol{\beta}}(s)$	A function mapping arclength to a tangent direction, used to define a $\mathbb{P}$.
$\mathbb{W}(\mathbb{P})$	A function mapping which computes the weight of a path $\mathbb{P}$.
$\mathbb{W}^R(\mathbb{P})$	A function mapping which computes the regularized weight of a path $\mathbb{P}$.
$\mathbb{W}^R_{\mathbb{S}}(\mathbb{S})$	A function mapping which computes the regularized weight of a segment $\mathbb{S}$.
pdf	Probability density function

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Electromagnetic radiation is the driving force behind many exciting and important phenomena in our universe. Many disciplines focus on narrower subsets of the full spectrum for solving specific problems. Medical disciplines focus on X-ray propagation for medical diagnosis purposes. The defense industry focuses on many subsets, including infrared for night vision capabilities and high-frequency portions of the spectrum for nuclear defense. In computer graphics, visible light dominates the portion of the spectrum studied in research that attempts to realistically simulate accurate propagation and physical interaction of light within virtual environments containing objects, volumetric mediums, and light emitters.

Since its first introduction to the field of computer graphics [1], volume rendering has been established as an important style of rendering for many disciplines: cinematic visual effects, video games, and the medical field, to name a few. For many of these environments, the key to realistic and believably-rendered images involves accurate simulation of visible light transfer through the scattering media. Although computer graphics disciplines focus on the simulation of the visible spectrum, volume radiative transfer solutions often extend naturally to other portions of the EM spectrum and particle transport. For this reason, general radiative transfer solutions in computer graphics tend to make good candidates for other scientific disciplines requiring radiative transfer modeling in infrared, visible, ultra-violet, remote sensing, and neutron transport domains.

The computer graphics community commonly focuses on finding solutions to the rendering equation [2]. Notably, in its original form, the rendering equation fails to capture the behavior of light scattering within a volume. However, extensions have been made to include the effects of volume scattering. This limits the use of many published solutions of light transfer to environments that do not contain volumes of suspended particles that scatter light moving through them. However, a better solution is needed since many important environments include volumes that often significantly impact the final light transport. To capture this behavior, graphics researchers have turned to neighbor fields and their radiative transfer model [3, 4], the equation of transfer [5]. These theories are fully unified in the path integral form of the light transport equation for the volume scattering case [6].

State-of-the-art research provides many reasonable solutions for the radiative transfer simulation, such as path-tracing, bi-directional path tracing, and photon mapping [7, 8, 9, 10]. In addition, basic radiative transfer models of volumes with emission only are fairly simple to solve, with techniques like ray-marching [11]. However, even with the introduction of just a single scattering event, solutions tend to become computationally complex quickly. With each additional scattering event, solutions must capture exponentially more possible paths for light to travel through a scene. As a result, solutions that attempt to model light transport through scattering volumes thoroughly tend to grow exponentially in computation time. This results in the various existing numerical schemes for multiple scattering tending to balance physical accuracy vs. increasing computation time. Often, the more computationally expensive of these methods still struggle to capture more than a few scattering events, despite their heightened cost.

One powerful model of radiative transfer formulates the problem as a Feynman Path Integral (FPI) [12, 13, 14]. Referred to as FPI-RT in the remainder of this work, this model is unique in its ability to capture a large number of scattering events within a volumetric medium, a notoriously difficult problem in computer graphics. Unlike existing solutions, FPIs are particularly interesting as the construction of the FPI-RT models all possible light transport through an environment simultaneously, effectively capturing all multiple scattering events within a volume. A previous numerical solver of the FPI utilized a discrete Frenet-Serret representation of paths in space [14]. In their representation, the resolution of the paths directly correlated to the maximum number of possible scattering events captured by the solver, with their method easily capturing upwards of 200 scattering events. While the existing numerical solution produced impressive results by proving that the FPI could be numerically evaluated, and despite the solver no longer scaling exponentially with the number of scattering events, the solver, unfortunately, remained computationally expensive, utilizing a time-consuming root-solver-based perturbation method for generating paths. This work was further optimized [13] to reduce the complexity of the root-solver by minimizing the number of operations per iteration and by introducing a Metropolis sampling technique. Still, the implementation remained slow because of the large number of paths required, an expensive representation of the paths, and a dependence on an expensive algorithm for generating paths.

Nevertheless, the FPI representation of radiative transfer remains of interest as it can feasibly compute results for many scattering events and its ability to generalize to any portion of the electromagnetic spectrum of interest. Thus, an efficient numerical solution would be particularly interesting to all fields desiring to simulate radiative transfer through a scattering environment. In particular, my goal is that with continued future work, my numerical solution of the FPI-RT can act as a baseline for performance comparisons of methods of radiative transfer focused particularly on high orders of scattering, as opposed to the commonly selected path trace methods, which are computationally expensive and severely limited in their ability to capture high orders of scattering.

This work presents novel numerical approaches for solving the FPI-RT, which perform significantly better than previous work. Additionally, I present a novel path generation numerical solver for the FPI-RT, validation for the solver, and a sample application of applying the solvers to render volumetric models. This work can be summarized as the following contributions.

1. In the first key contribution, I present a path perturbation numerical solver using a simple yet powerful representation of paths and an efficient path perturbation algorithm for numerically computing the FPI-RT. Additionally, I show that while numerically stable and allowing both an efficient CPU and GPU implementation, this optimized path-perturbation numerical solver is limited by bias introduced by the path-perturbation approach. Lastly, I discuss the limitations of the path perturbation approach, leading to my next contribution.

2. In a second key contribution, I present and verify a novel numerical solution approach based

on path generation, in which the solver randomly explores valid path space. I validate this solver by comparing it against other numerical solvers for a simplified weighting function, allowing for an analytical derivation of the simplified weight path integral (FPI-SW) solution for small segment count paths. Additionally, I verify using the full radiative transfer weighting kernel. Together, these two novel solvers bring the benefits of the FPI formulation, most notably the ease in naturally modeling high orders of scattering, to a realistic computation time of under an hour.

3. In a last contribution, I demonstrate that the proposed path generation solver is powerful enough to render images within simple environments. While this demonstration falls in the domain of computer graphics, the application of the solver is not limited to the visible light portion of the electromagnetic spectrum, and the method could easily simulate other portions of the spectrum for different research domains.

The rest of the document continues as follows. Chapter 2 summarizes related work on radiative transfer simulation and multiple scattering, providing separate discussions for single and multiple scattering domains. Chapter 3 provides a detailed discussion of the FPI formulation of radiative transfer and introduces the most powerful prior path perturbation numerical solver. Chapter 4 introduces a novel path perturbation numerical solver for the FPI-RT and discusses the method's limitations. Chapter 5 introduces a simplified weighting kernel, an analytical derivation of the simplified weight FPI, FPI-SW, for small segment path counts, and a novel path-generation numerical solver for FPI-SW and FPI-RT. Chapter 6 then presents the application of the novel path-generation numerical solver of FPI-RT to render images of different volumes within a virtual environment. Finally, Chapter 7 summarizes all findings and presents some directions for future work.

2. RELATED WORK

Existing methods that solve volumetric scattering of radiative transfer within a medium can be split into two categories based on the number of scattering events in the model. The first category contains those methods that capture only a single scatter event, visualized in Figure 2.1. The second category contains methods that capture more than one (i.e. multiple) scattering events, visualized in Figure 2.2. In general, methods that capture higher numbers of scatter events are more accurate as they capture more light transfer within a scene. However, methods that capture higher numbers of scattering events tend to be more computationally complex, thus preventing their use in many situations with strict time budgets.

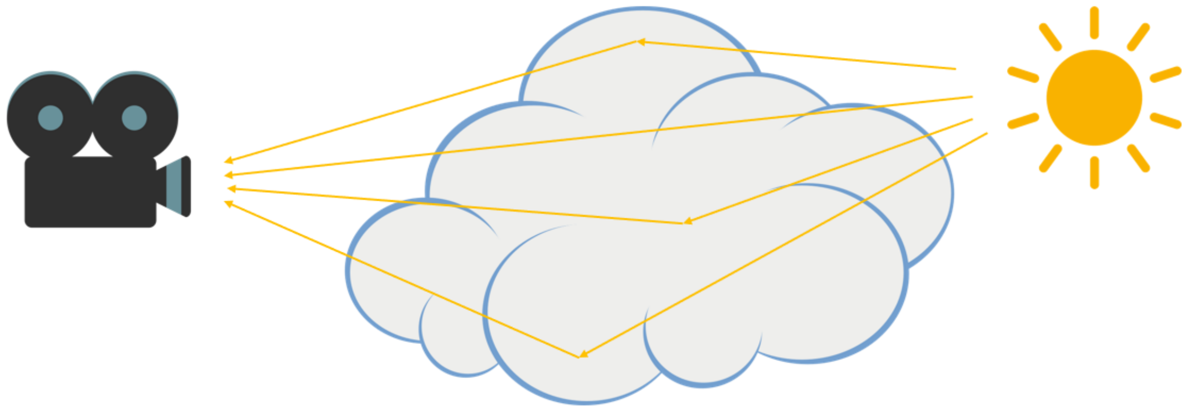


Figure 2.1: Visualization of single scattering within an environment. Light traveling from an emitter to a receiver can scatter, at most, one time.

2.1 Single Scattering Methods

Single scattering methods capture only one indirect scatter of emitted light within a volume before detection by a receiver. While several methods exist for solving the single-scatter case, all are based

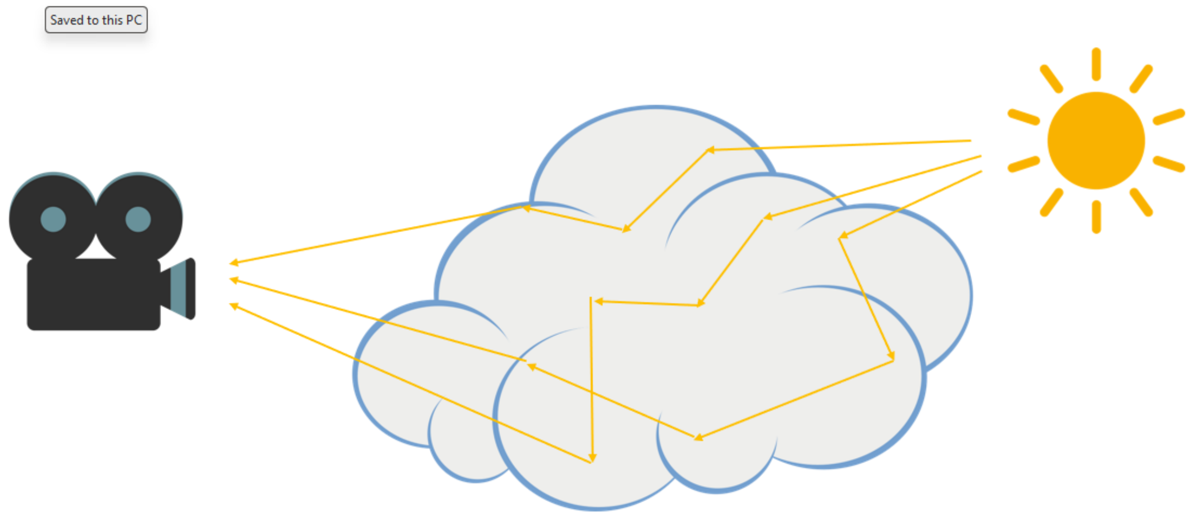


Figure 2.2: Visualization of multiple scattering within an environment. Unlike single-scattering environments, light traveling from a source to a receiver can scatter (change direction) any number of times. More advanced techniques aim to capture higher orders of this scattering.

on this strong limiting assumption that light can scatter only a single time in a volume media and thus are impractical for capturing more realistic radiative transfer through a real physical medium. Additionally, most proposed solutions lack generality, targeting a single specific type of media such as clouds or planetary rings only, and thus are not applicable for general use.

Blinn's notable single scattering solution attempted to visualize the rings of Saturn and other planet atmospheres [1]. While providing an analytical solution, this method is severely limited in its assumption that a light source and the viewer are located at infinity. Kajiya and Von Herzen [7] extended Blinn's method for ray tracing and eliminated the viewer and light placement at infinity restrictions. While this prevents an analytic solution, this method is generalized to work in various environments. Finally, Ebert and Parent implemented this model in the scan line context, allowing the utilization of hardware acceleration [15]. Although these three techniques together represent a general model for capturing single scatter events, they do not apply to most volumetric media.

While these single scattering techniques have their applications, they are quite limited in the total scattering captured compared to the solutions I propose in this work. Thus I limit my discussion of these techniques and defer to a survey paper [16] for a deeper discussion of other single scattering techniques.

2.2 Multiple Scattering Methods

The earliest numerical solution for radiative transfer in computer graphics was by Kajiya and von Herzen [7]. This method traces paths through the volume, and scattering and absorption events are directly simulated during path construction. Unfortunately, this method suffers from a high computation cost, notably due to the many paths required for convergence. Several other techniques directly improve that solution. Lafortune and Williams [8] modify the method to use bidirectional construction, greatly improving the convergence of the method by constructing paths starting from both the emitter and the receiver. Pauly et al. [9] introduce the Metropolis-Hastings algorithm, also improving the convergence of the original path sampling method by biasing the solver toward paths that contribute most toward the final render (i.e. transfer the most light). While many additional works introduce multiple-importance sampling, volumetric density estimation, and point and beam estimators, Krivanek et al. [17] unify all the techniques into a single model capable of high-quality renders. Bitterli and Jarosz [18] take this one step further by generalizing the existing 0-dimensional photon points and 1-dimensional photon beams into n-dimensional samples, improving the run-time of existing volumetric photon mapping techniques by a factor of 2.4x to 40x. These photon or path tracing methods are the most flexible, easily incorporating new scattering functions and volume representations. However, despite many improvements and approaches over the years, these models remain too slow for many real-world applications.

Another family of multiple scattering techniques starts with the assumption that a material has a high enough albedo, or scattering likelihood, that light traveling through the volume can be modeled as experiencing isotropic scattering within the medium due to a high number of scattering events. In these models, diffusion theory is used to model the radiative transfer within the medium

[19]. Some works have increased the speed of these diffusion techniques via directional dipole models or multiple monopoles [20, 21]. These techniques approximate scattering as the net effect of one or two point lights placed on either side of a medium's boundary, providing semi-analytic solutions that rely on a separate Monte Carlo tracing step for single-scattering or directly including the single-scattering in a complete analytic solution. While these methods are incredibly useful for high albedo materials, such as skin, these diffusion techniques are poor models for volumes with a mean free path, the average distance a particle travels before experiencing a scattering event, closer to the length of the medium (such as clouds).

Many techniques shift a significant amount of their solution to a pre-calculation step. Szirmay et al. pre-computes several global light paths through a static volume [22]. Then, during the actual runtime of the method, this global pre-computed light path structure is queried and used to calculate transport through the volume at interactive frame rates. While the camera and light source are free to move during runtime, the volume must remain static, limiting its use in real-world environments. However, this technique does achieve interactive frame rates. Lee and O'Sullivan [23] pre-compute a set of probability distributions within a homogeneous volume. During the evaluation, while standard path tracing is used near the surface of the volume, deeper portions of the path utilize the pre-computed functions to sample path traversal within the medium. Mueller et al. [24] adapt this method to support heterogeneous environments. While these techniques can achieve interactive frame rates, they struggle to generate physically accurate images.

A recent alternative to rendering multiple scattering effects is Disney Research's Deep Scattering method [25]. In this work, the authors train a neural network on a large database of clouds to learn the spatial and directional radiant flux through the training environments. Then, for final network evaluation, the network is fed a hierarchical grid of samples constructed to best sample the heterogeneous density within the cloud structure. While the network is trained to predict multiple scattering, all direct lighting and single scattering events are calculated using standard Monte Carlo path tracing. The main strength of this method is its ability to capture the quality of the ground truth images within minutes.

Additionally, the method can accurately capture different scattering functions if the network is trained on ground truth images using the desired scattering method. However, the authors, unfortunately, fail to discuss the number of scatters captured utilizing this method. Thus a direct analysis in comparison to the FPI method is not possible. The number of captured scatters is likely directly related to the method utilized to generate the training data set. Thus the method is limited to, at best, capturing the number of scattering events represented in the test data set.

2.3 Feynman Path Integral Construction of Radiative Transfer

A notably different approach to solving radiative transfer is through a Feynman path integral formulation, first constructed by Tessendorf [12]. In this model, drawing from the field of quantum mechanics, the probability of each potential path of travel of a photon is calculated using a Feynman path integral [26], and these are summed to capture the overall probability density of the photon traveling from a starting position to an ending position. Unlike the other multiple scattering methods, this method is constructed to capture an arbitrary number of scattering events, easily capturing significantly more scatter events than even the best of other existing multiple scattering methods. For environments where exact, physically accurate calculations are required, the FPI construction is an extremely powerful technique. While the original formulation [12] was constrained to slab geometry and ignored backscatter, Tessendorf later derived a more general time-dependent representation, allowing the use in environments with complex geometry and with support for backscattering [27]. This work was continued by Tessendorf and Kilgo in their construction of a numerical solution directly integrating the FPI in its time-dependent form [14, 13, 28]. While they demonstrate the ability of their solver to converge and accurately fit measured beam spread data [13], their method has some limitations. Firstly, the FPI construction requires the evaluation of a normalizing term that is currently computationally prohibitive to evaluate and thus was ignored. Additionally, the method is slow, taking compute years to generate a significantly converged solution.

The Feynman path integral also requires a Fourier-transformed representation of a volume medium's phase function. While Tessendorf [14] introduces a Gaussian Fourier phase function, the incorporation of other notable phase functions is limited by the need to derive each phase function's Fourier transform. Lastly, the direct integration of the FPI is computationally expensive, on the order of multiple compute years, limiting its use in real-world applications in the current form.

A recent work by Petrzala [29] challenges the work of Tessendorf [12, 27, 14], claiming that the presented derivation fails to measure up to its claims due to two critical errors: a failure to capture an overall normalizing component and an incorrect assumption that a two-dimensional delta function can be treated as a three-dimensional delta function. Additionally, Petrzala presents a probabilistic approach to solving the FPI for an optically thick environment with a sharply peaked Gaussian phase function, providing a different approach to solving the FPI.

Some other works explore the Feynman path integral approach to radiative transfer. Perelman et al. apply the FPI to the problem of light propagation in turbid media [30, 31]. These works focus on finding the most probable path for photon transport through a medium to represent the overall photon transfer. However, these techniques are limited to optically thick media and limit the scattering to small angles only.

Wilson and Wang apply the Feynman path integral to the problem of infrared light scattering within tissue [32]. Due to tissue material's high albedo for infrared radiation, a numerical solution of infrared transfer through tissue requires a model that captures high scattering orders. Wilson and Wang model scattering as a Lagrangian, which captures albedo and scattering, integrated along the path as an energy function. This function is minimized to obtain the most likely path. Their method can explicitly model absorption and respect constraints on intermediate points, directions, or path length.

Premoze et al. [33] also present an approach for solving the FPI by identifying the most probable path of light in the medium. This method evaluates the path integral like the WKB (Wentzel-

Kramers-Brillouin) expansion by assuming that the most probable path and only its closest neighbors can approximate the full transfer. This is a form of variance reduction compared to other Monte Carlo techniques that must discover the highest weighted paths. While efficient, this assumption limits the solver's use to environments whose less likely photon paths have an insignificant impact on the final transfer. Additionally, this method assumes a uniform density volume and a strong forward scattering phase function.

While many FPI techniques and their approximations allow for faster FPI-RT simulation within specific environments, they lack the general applicability of the original formulation. However, the existing numerical solver of the original FPI-RT formulation remains computationally prohibative for use. In contrast, my presented numerical solvers maintain the generality of the full FPI-RT construction while significantly improving overall computation efficiency (an improvement of four orders of magnitude).

3. FEYNMAN PATH INTEGRAL FORMULATION of RADIATIVE TRANSFER

My work focuses on efficiently computing the Feynman path integral formulation of radiative transfer [12], and is heavily influenced by the work of both Tessendorf and Kilgo [14, 13]. In this chapter, I provide a summary of radiative transfer, the FPI formulation of radiative transfer (FPI-RT), and describe a previously published path perturbation numerical solver.

3.1 The Rendering Equation and Radiative Transfer

I begin with a discussion of the rendering equation [2], eq 3.1, which following the geometric optics approximation captures all light emitted and scattered throughout a scene by surfaces, both directly and indirectly.

$$\begin{aligned} L_o(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \lambda, t) &= L_e(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \lambda, t) \\ &\quad + \int_\Omega f_r(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_S, \hat{\boldsymbol{\omega}}_R, \lambda, t) L_i(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_S, \lambda, t)(\hat{\boldsymbol{\omega}}_S \cdot \hat{\vec{\boldsymbol{n}}}) d\hat{\boldsymbol{\omega}}_S \end{aligned} \tag{3.1}$$

This equation represents the light detected at position $\vec{\boldsymbol{x}}_R$ from direction $\hat{\boldsymbol{\omega}}_R$ as a combination of the following:

1. Light emitted at $\vec{\boldsymbol{x}}_R$ in direction $\hat{\boldsymbol{\omega}}_R$, $L_e(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \lambda, t)$

2. The integration over the surface at point $\vec{\boldsymbol{x}}_R$ from all incoming directions $\hat{\boldsymbol{\omega}}_S$ from position $\vec{\boldsymbol{x}}_S$. In this, f_r represents the reflectance function capturing the amount of light coming in from $\vec{\boldsymbol{x}}_S$ in direction $\hat{\boldsymbol{\omega}}_S$ which is reflected in $\hat{\boldsymbol{\omega}}_R$. Additionally, this term includes a cos weighting factor between the incoming direction $\hat{\boldsymbol{\omega}}_S$ and surface normal $\hat{\vec{\boldsymbol{n}}}$.

In its original form, the rendering equation ignores a number of physical effects that have minimal impact on the final image, such as light polarization, the Doppler effect, and phosphorescence; the radiative transfer model I use is no different and ignores the effects as well. Additionally, many computer graphics models further simplify two additional terms. First, the long-term steady-state assumption is made, removing the dependence on t, as it is assumed that the light in an environment that has reached a steady state. Additionally, while the general form of the rendering equation is parameterized on the wavelength of light in question λ, this is often simplified to three channels, red, green, and blue. Again, the models I focus on in this work also make these two assumptions, focusing on a single wavelength of light at a time, which I simply refer to as the intensity of emitted light, as well as focusing on the time-independent steady state of light.

While incredibly powerful, the rendering equation in this original form does not capture the effect of volumes on the light in a scene. While the equation has been extended to the volumetric domain [2], to fully capture the interaction of light with suspended particles in a virtual environment, most computer graphics researchers instead turn to the radiative transfer equation (RTE), eq 3.2 [34]. The radiative transfer equation closely parallels the rendering equation, modeling the behavior of light within a virtual environment. However, in addition to modeling the interaction of light bouncing between surfaces, the radiative transfer equation additionally captures the physical effects of absorption, scattering, and emission of electromagnetic radiation; all of which are physical properties of scattering media that significantly impact light energy propagating through a medium. I describe these effects through the lens of each interaction's effect on a small change in radiance $L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R)$ propagating in direction $\hat{\boldsymbol{\omega}}_R$ at position $\vec{\boldsymbol{x}}_R$.

$$\begin{aligned}(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\; L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = &-(\sigma_a + \sigma_b) L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) \\ &+ Q(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) + \sigma_b \int_{\Omega} L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_S)\; P d\hat{\boldsymbol{\omega}}_R\end{aligned} \tag{3.2}$$

The left side of this equation, $(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\; L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R)$ represents the time and position dependent change in $L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R)$, where $L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R)$ represents the radiance at position $\vec{\boldsymbol{x}}_R$ in direction $\hat{\boldsymbol{\omega}}_R$. The terms of the right side of the equation are described next.

Absorption represents the conversion of light energy to another form of energy, usually heat, resulting in a net reduction in the intensity of propagating radiation, as modeled by eq 3.3. Examples of the effect of absorption include the heating of a puddle in the sun enough to trigger evaporation, or the inability to see through smoke due to the suspended carbon participates' absorption of light.

$$(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\, L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = -\sigma_a\, L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) \tag{3.3}$$

Scattering is the redirection of propagating electromagnetic radiation from an original direction to different propagation directions. This manifests in two important ways, out-scattering describes the reduction of light traveling in direction $\hat{\boldsymbol{\omega}}_R$ due to scattering into a new direction $\hat{\boldsymbol{\omega}}_S$ when interacting with the media, modeled by eq 3.4.

$$(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\, L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = -\sigma_b\, L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) \tag{3.4}$$

In contrast, in-scattering describes the scattering of light from other directions $\hat{\boldsymbol{\omega}}_S$ into direction $\hat{\boldsymbol{\omega}}_R$ and thus is an increase of radiance, as seen in eq 3.5. In modeling in-scattering, I make use of a phase function P, which represents the normalized distribution of intensity distributed from $\hat{\boldsymbol{\omega}}_S$ in the direction of $\hat{\boldsymbol{\omega}}_R$. The important effect of scattering is responsible for the sky appearing blue, as well as the inability to see more than a few feet into dense fog, with the suspended vapor's highly impactful scattering of light as it makes its way to a receiver.

As in-scattering at a position $\vec{\boldsymbol{x}}_R$ potentially depends on all incoming light, I must integrate over all possible incoming light directions $\hat{\boldsymbol{\omega}}_S$. Additionally, the phase function must be normalized, as shown in 3.5.

$$
\begin{gathered}
(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\ L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = \int_{\Omega'} P\ L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_S) d\hat{\boldsymbol{\omega}}_S \\
\int_{\Omega} P d\hat{\boldsymbol{\omega}}_S = 1
\end{gathered}
\tag{3.5}
$$

Lastly, emission models the conversion of one energy type, often nuclear in the case of stars or heat in the case of black bodies, to electromagnetic radiation, causing effects like electric stove tops glowing red as their temperature increases. Emission is commonly modeled as a source term, $Q(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R)$, with the full form shown in eq 3.6.

$$
(\hat{\boldsymbol{\omega}}_R \cdot \vec{\nabla})\ L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = Q(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) \tag{3.6}
$$

Additionally, it is helpful to write the RTE in an integral form rather than the integro-differential form previously shown in Equation 3.2. In this new formulation, shown as Equation 3.7, radiance at a single point $\vec{\boldsymbol{x}}_R$ in direction $\hat{\boldsymbol{\omega}}_R$ is modeled as the combined contribution of all transported light from all positions and directions in the long-term time limit. In this, I integrate over all possible arclengths of light paths $\int_0^\infty ds$, all possible starting positions of light in the environment $\int_V d^3\vec{\boldsymbol{x}}_S$, and all possible start directions $\int_\Omega d\hat{\boldsymbol{\omega}}_S$. In the next section, I will describe the construction of $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ in the path integral formulation.

$$
L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = \int_0^\infty ds\ \int_V d^3\ \vec{\boldsymbol{x}}_S\ \int_\Omega d\hat{\boldsymbol{\omega}}_S\, G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)\ Q(\vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S) \tag{3.7}
$$

3.2 Feynman Path Integral Formulation of Radiative Transfer

The Feynman path integral (FPI) was introduced by Feynman as a tool for computing the probability amplitude of a quantum particle [26]. In the standard construction of a path integral, a transition matrix representing the likelihood of a path to be taken by a particle is applied to all possible paths a quantum particle could take through an environment. Afterward, the weighted paths are summed with their total representing the overall probability of the particle undergoing the transition in question. Tessendorf [12] formulated the radiative transfer problem to pose it as a path integral with a transport kernel $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, which has an explicit representation in terms of a Feynman path integral:

$$
\begin{aligned}
G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = e^{-\sigma_t S} \int D\mathbb{P} \int Dp\, \delta\left(\hat{\boldsymbol{\beta}}\left(0\right) - \hat{\boldsymbol{\omega}}_S\right)\, \delta\left(\hat{\boldsymbol{\beta}}\left(s\right) - \hat{\boldsymbol{\omega}}_R\right) \\
\times\, \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right)\, \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_S\right) \\
\times\, \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S ds' \hat{\boldsymbol{\beta}}\left(s'\right)\right) \\
\times \exp\left(\sigma_b \int_0^S ds'\, \widetilde{\boldsymbol{Z}}\left(\vec{\boldsymbol{p}}\left(s'\right)\right)\right) \\
\times \exp\left(i \int_0^S ds'\, \vec{\boldsymbol{p}}\left(s'\right) \cdot \frac{d\hat{\boldsymbol{\beta}}\left(s'\right)}{ds'}\right)
\end{aligned}
\tag{3.8}
$$

In this, $D\mathbb{P}$ represents an integration over valid path space, while Dp represents an integration over the Fourier variable $\vec{\boldsymbol{p}}$. Each valid path $\mathbb{P}$ is defined by an arclength parameterized curve $\vec{\boldsymbol{r}}(s)$ mapping arclength to a world space position. $\hat{\boldsymbol{\beta}}(s)$ likewise maps arclength to the tangent of the curve, such that $\hat{\boldsymbol{\beta}}(s) = (d\vec{\boldsymbol{r}}(s)/ds)$.

In this path integral construction, the radiative transfer through the environment can be seen as an integration over $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$. While its complete form is presented here, a derivation of the transport kernel, $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ is presented in Appendix A. Plugging this into the integral formulation of radiative transfer, Equation 3.7, I obtain that radiance $L\left(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R\right)$ arriving at a position $\vec{\boldsymbol{x}}_R$ in the direction $\hat{\boldsymbol{\omega}}_R$, given the radiance source $L_0\left(\vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, is

$$L\left(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R\right) = \int_\Omega d\hat{\boldsymbol{\omega}}_S \int_V d^3\vec{\boldsymbol{x}}_S \int_0^\infty dS\, G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)\, Q\left(\vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) \tag{3.9}$$

In the full radiative transfer solution of light received $L\left(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R\right)$, I must perform the full integration over all possible starting positions $\vec{\boldsymbol{x}}_S$, starting directions $\hat{\boldsymbol{\omega}}_S \in \Omega$, and path lengths S. While this will be important to revisit when I apply the FPI formulation to render an image within a virtual environment, most of my discussion of prior and novel numerical solver approaches will focus specifically on computing a single, fixed transport kernel, $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$.

Evaluation of single path integral transport kernel $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ focuses only on paths that meet a specific set of boundary condition parameters, $\mathbb{R} = S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S$. For this specific evaluation, an arclength-parameterized curve $\mathbb{P}$ is modeled as a position function of arclength, $\boldsymbol{r}(s)$, and tangent unit vector function of arclength $\hat{\boldsymbol{\beta}}(s) = d\boldsymbol{r}(s)/ds$. For a given path $\mathbb{P}$ to be considered valid, it must meet a number of conditions specified by the boundary constraints $\mathbb{R}$.

1. The path $\mathbb{P}$ must be of specific arclength S.

2. Path $\mathbb{P}$'s position function $\boldsymbol{r}(s)$ must be formed such that, given the boundary constraint specified starting and ending positions of the path integral, $\vec{\boldsymbol{x}}_S$ and $\vec{\boldsymbol{x}}_R$, $\vec{\boldsymbol{r}}(0) = \vec{\boldsymbol{x}}_S$ and $\vec{\boldsymbol{r}}(S) = \vec{\boldsymbol{x}}_R$.

3. Path $\mathbb{P}$'s tangent function $\hat{\boldsymbol{\beta}}(s)$ must be formed such that, given the boundary constraint specified starting and ending directions of the path integral, $\hat{\boldsymbol{\omega}}_S$ and $\hat{\boldsymbol{\omega}}_R$, $\hat{\boldsymbol{\beta}}(0) = \hat{\boldsymbol{\omega}}_S$ and $\hat{\boldsymbol{\beta}}(S) = \hat{\boldsymbol{\omega}}_R$.

In eq 3.8, one can see that the five delta functions enforce these specified boundary conditions.

Lastly, I specify that $\vec{\boldsymbol{p}}(s)$ represents the Fourier functional integration variable at arclength parameter s, and $\widetilde{\boldsymbol{Z}}(p)$ represents the Fourier transform of the medium's phase function P. While this seems an unnatural choice to those in the computer graphics domain, this approach is standard in physics. It allows for the full derivation of the weighting function, as shown in Appendix A.

Lastly, this construction defines a few functions representing the environment $\mathbb{E}$ through which the FPI is calculated. In the general case, the environment is defined as two functions that can vary spatially; $\sigma_a(\vec{\boldsymbol{x}})$ represents the absorption at position $\vec{\boldsymbol{x}}_R$, and $\sigma_b(\vec{\boldsymbol{x}})$ represents the scattering at position $\vec{\boldsymbol{x}}_R$. Additionally, it is convenient to combine absorption and scattering into a single function $\sigma_t(\vec{\boldsymbol{x}}) = \sigma_a(\vec{\boldsymbol{x}}) + \sigma_b(\vec{\boldsymbol{x}})$ representing the extinction of the environment at position $\vec{\boldsymbol{x}}$.

3.3 Prior Numerical Solver of Feynman Path Integral Radiative Transfer

While deriving an exact, analytical solution of the FPI formulation of radiative transfer remains infeasible for all but the simplest of cases, prior work exploring a numerical solution has been published which utilizes Monte Carlo techniques to generate and weight a subset of the total path set [14]. This previous numerical solver calculates the path integral $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ through the method shown in Algorithm 1.

Algorithm 1 Frenet Serret Path Perturbation Numerical Solver

```
Input: $\mathbb{R} = \{S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\}$
Output: $I$
$\mathbb{P}_0 \leftarrow GenerateSeedCurve(\mathbb{R})$
$pathsSet \leftarrow \emptyset$
while $||pathsSet|| < N$ do
    $\mathbb{P}_{i+1} = PerturbPath(\mathbb{P}_i)$
    $pathsSet \leftarrow pathsSet \cup PerturbPath(\mathbb{P}_{i+1})$
end while
$I \leftarrow 0$
for $\mathbb{P}_i \in pathsSet$ do
    $I \leftarrow I + \mathbb{W}^R(\mathbb{P}_i)$
end for
$I \leftarrow I/N$
return $I$
```

In this prior numerical solver, an initial path is generated by constructing a simple, five-control point Bezier curve such that the start and end positions and normals, as well as the total arclength specified in $\mathbb{R}$, were maintained. However, as this initial seeding method is computationally expensive and extremely limited in the geometry of paths it could generate, the authors decided that path generation would be utilized only to generate the first path to seed the numerical solver. To circumvent the limitations of the expensive path generation, all other paths were generated by perturbing previously generated paths using a faster and non-biased perturbation algorithm [14].

3.3.1 Discrete Path Representation

For their numerical solution, the prior work's authors required a discretized representation of paths. Thus, each path in their numerical solver is represented via a discrete Frenet-Serret framework, where the Frenet-Serret frame is a specific construction of an orthonormal frame for curves [35].

The path representation is as follows. Given a path $\mathbb{P}$'s arclength-parameterized curve $\boldsymbol{r}(s)$, three Frenet-Serret vector functions, Tangent $\hat{\boldsymbol{T}}(s)$, Normal $\hat{\boldsymbol{N}}(s)$, and Binormal $\hat{\boldsymbol{B}}(s)$ are defined as shown in eq 3.10, forming a local orthonormal basis.

$$\begin{aligned}
\hat{\boldsymbol{T}}(s) &= \frac{d\boldsymbol{r}(s)}{ds} \\
\hat{\boldsymbol{N}}(s) &= \frac{d\hat{\boldsymbol{T}}(s)}{ds} \Big/ \left|\frac{d\hat{\boldsymbol{T}}(s)}{ds}\right| \\
\hat{\boldsymbol{B}}(s) &= \hat{\boldsymbol{T}}(s) \times \hat{\boldsymbol{N}}(s)
\end{aligned} \tag{3.10}$$

Additionally, these vector functions are related to the curvature $\kappa(s)$ and torsion $\tau(s)$ of the curve via a first order differential equation:

$$\begin{bmatrix} d\hat{\boldsymbol{T}}(s)/ds \\ d\hat{\boldsymbol{N}}(s)/ds \\ d\hat{\boldsymbol{B}}(s)/ds \end{bmatrix} = \begin{bmatrix} 0 & \kappa(s) & 0 \\ -\kappa(s) & 0 & \tau(s) \\ 0 & -\tau(s) & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{T}}(s) \\ \hat{\boldsymbol{N}}(s) \\ \hat{\boldsymbol{B}}(s) \end{bmatrix} \tag{3.11}$$

Utilizing these properties, the prior numerical solver divides each path into M discrete segments with arclength $\Delta s = S/M$ [14]. Each segment $\mathbb{S}_i, 0 \leq i < M$ has an associated curvature and torsion value, κ_i, τ_i, which allows the construction of a matrix per segment $\boldsymbol{U}_i(\kappa_i, \tau_i)$ representing the rotation matrix transforming the Frenet frame at the starting point of the segment to the ending point of the same segment. In addition to its segments, each path $\mathbb{P}$ has an associated initial Frenet-Serret reference frame $\boldsymbol{F}_0$, as well as starting position $\vec{\boldsymbol{x}}_S$ which is enforced to meet the initial position constraint in $\mathbb{R}$. Likewise, the initial frame $\boldsymbol{F}_0$ is constructed such that its tangent value $\hat{\boldsymbol{T}}_0$ is enforced to be $\hat{\boldsymbol{\omega}}_S$. The stored base reference frame $\boldsymbol{F}_0$ and per segment matrices U_i reconstruct the position and reference frame of any segment as

$$\vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_{i-1} + \hat{\boldsymbol{T}}_i \, \Delta s \qquad\qquad \boldsymbol{F}_i = \boldsymbol{U}_i * \boldsymbol{F}_{i-1} \tag{3.12}$$

where the segment matrices are

$$\boldsymbol{U}_i = \begin{bmatrix} 1 - (\kappa_i^2/l_i^2)\,(1 - \cos{(l_i \Delta s)}) & (\kappa_i/l_i)\sin{(l_i \Delta s)} & (\kappa_i \tau_i/l_i^2)\,(1 - \cos{(l_i \Delta s)}) \\ -(\kappa_i/l_i)\sin{(l_i \Delta s)} & \cos{(l_i \Delta s)} & (\tau_i/l_i)\sin{(l_i \Delta s)} \\ (\kappa_i \tau_i/l_i^2)\,(1 - \cos{(l_i \Delta s)}) & -(\tau_i/l_i)\sin{(l_i \Delta s)} & 1 - (\tau_i^2/l_i^2)\,(1 - \cos{(l_i \Delta s)}) \end{bmatrix} \tag{3.13}$$

$$l_i = \sqrt{\kappa_i^2 + \tau_i^2} \tag{3.14}$$

3.3.2 Root-Solve Path Perturbation

In the prior work solver, a root-finder perturbation approach is utilized to generate a novel path $\mathbb{P}'$ from an initial path $\mathbb{P}$, shown in Algorithm 2. First, three segments are selected along the curve $\mathbb{S}_l, \mathbb{S}_m, \mathbb{S}_r$ such that $l < m < r$. Note, each segment likewise has a torsion and curvature value, giving us six parameters $\kappa_l, \kappa_m, \kappa_r, \tau_l, \tau_m, \tau_r$. To generate a new curve, κ_m is randomly modified. Then, a root solve is performed over the remaining five parameters $\mathbb{K}$ to bend the newly generated path to meet the boundary constraints $\mathbb{R}$. For this, I solve the set of equations shown in Equation

3.15 for $RS(K) = 0$. Note that I use Equation 3.12 to calculate the curve's final position $\vec{\boldsymbol{x}}_M$ and the final tangent of the curve $\hat{\boldsymbol{T}}(S)$.

Algorithm 2 Root-Solver Path Perturbation

```
Input: $\mathbb{P}$
Output: $\mathbb{P}'$
$\mathbb{P}' \leftarrow \mathbb{P}$
$S \leftarrow AccessSegments(\mathbb{P})$
$l \leftarrow Rand(0, M-3)$
$m \leftarrow Rand(l+1, M-2)$
$r \leftarrow Rand(m+1, M-1)$
$S[m].\kappa \leftarrow Rand(\kappa_{min}, \kappa_{max})$
$K \leftarrow S[l].\kappa, S[r].\kappa, S[l].\tau, S[m].\tau, S[r].\tau$
$i \leftarrow 0$
while $i \leq maxSteps$ do
    $StepSolver(\mathbb{P}', \mathbb{K})$
    if $Converged(\mathbb{P}', \mathbb{K})$ then
        return $\mathbb{P}'$
    end if
end while
return
```

$$\mathbb{K} = \begin{bmatrix} \kappa_l & \kappa_r & \tau_l & \tau_m & \tau_r \end{bmatrix}^T \qquad RS(\mathbb{K}) = \begin{bmatrix} \vec{\boldsymbol{x}}_M[x] - \vec{\boldsymbol{x}}_R[x] \\ \vec{\boldsymbol{x}}_M[y] - \vec{\boldsymbol{x}}_R[y] \\ \vec{\boldsymbol{x}}_M[z] - \vec{\boldsymbol{x}}_R[z] \\ ||\hat{\boldsymbol{T}}(S) - \hat{\boldsymbol{\omega}}_R||_1 \\ ||\hat{\boldsymbol{T}}(S) - \hat{\boldsymbol{\omega}}_R||_2^2 \end{bmatrix} \tag{3.15}$$

If a set of parameters $\mathbb{K}$ can be found such that a solution Equation 3.15 is found, a new path $\mathbb{P}'$ has successfully been generated. However, this method is not guaranteed to converge and thus can fail to generate a new path. To handle these cases, an upper limit of optimization iterations is set.

If a path perturbation ever reaches this max iteration count without converging, the new path is discarded, and the perturbation process begins again from the initial curve.

3.3.3 Path Weighting

Lastly, once a set of valid paths have been generated, the full FPI solution can be calculated by weighting each path $\mathbb{P}_i$ and combining their weights into the final path integral solution.

$$\begin{aligned} G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) &= \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \frac{W(\mathbb{P}_i)}{pdf\left(\mathbb{P}_i\right)} \\ &= \lim_{N\to\infty} \mathbb{D} \frac{1}{N} \sum_{i=1}^{N} W(\mathbb{P}_i) \end{aligned} \tag{3.16}$$

$$W(\mathbb{P}_i) = \prod_{n=0}^{M-1} W_{\mathbb{S}}^{R}(\mathbb{S}_n) \tag{3.17}$$

$$W_{\mathbb{S}}^{R} = \exp\left(-\sigma_t \Delta s\right) \int_0^{\infty} \frac{p}{2\pi^2} \exp\left(\sigma_b \Delta s \; \widetilde{\boldsymbol{Z}}\left(p\right) - \frac{\epsilon^2}{2} p^2\right) \frac{\sin\left(p \kappa_n \Delta s\right)}{\kappa_n \Delta s} dp \tag{3.18}$$

In this formulation, $\mathbb{D}$ is a normalizing factor containing a number of factors that have typically been ignored, although I will address this more in the coming sections. For now, take note that the Monte Carlo pdf weighting has been combined into this normalization factor $\mathbb{D}$. $\mathbb{W}(\mathbb{P}_i)$ represents the weight of the i-th path and $\mathbb{W}_{\mathbb{S}}^{R}(\mathbb{S}_n)$ is the regularized weight of a single segment $\mathbb{S}_n$ of the path $\mathbb{P}_i$.

$\widetilde{\boldsymbol{Z}}(p)$ is defined to be the Fourier transformed phase function, discussed in the previous work [14]. In addition, previous work shows that for $\epsilon \neq 0$, with ϵ being a regularization parameter, the behavior of $\mathbb{W}_{\mathbb{S}}^{R}$ is numerically stable; as $\epsilon \to 0$, $W_{\mathbb{S}}^{R}$ converges to the exact, non-regularized weight of a single path segment. Again, while their final forms are shown here, a full derivation of the Feynman path integral radiative transport kernel $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, as well as path and segment weight functions and their regularized forms, can be found in Appendix A.

In an important optimization, in a homogenous environment, the weight of a path depends only on the curvature, number of segments in the path, and total path arclength. This allows a table of segment weights to be precalculated for curvature values independent of the paths generated within the environment for a wide range of possible curvature values. This significantly improves the run time of my method, allowing multiple FPI-RTs to be calculated within one virtual environment, all sharing the precalculated table. Unfortunately, this assumption breaks down when scattering and absorption parameters are non-homogenous, requiring a world space positional lookup per segment to determine the local scattering and absorption parameters. Each scattering and absorption paramter combination requires a unique precalculated weight table, with the total memory footprint of precalculated weight tables scaling quickly with environment complexity. For now, I mitigate this by only allowing a small subset of possible scattering and absorption paramter combinations within a single environment, allowing for precalculating a weight table for each of these few combinations. However, a new approach is likely needed as more complex environments are explored in future work.

Summarizing the prior work numerical solver, to calculate $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, N total paths are generated; an initial curve $\mathbb{P}_0$ is generated using an expensive, Bezier-based method, after which existing paths are continually perturbed until N unique paths $\mathbb{P}$ exist. Finally, all weights are determined for each path $\mathbb{P}$, with their combined weight representing a single path integral solution, I.

3.4 Limitations of Prior Work

The previously published, root-solver numerical solver is limited in two primary ways. First, the solver utilizes an expensive representation of paths, selected for its strict maintenance of path length and explicit storage of curvature along the discrete path. This requires the storing of curvature and torsion values for each segment along the curve, as well as expensive matrix rotation transforms and updates for each segment along the curve. Secondly, the selected path representation led to the selection of an expensive, and unbound perturbation algorithm based on direct manipulation of the Frenet-Serret parameters. While this root-solve method was able to generate novel paths from existing paths that maintain the end constraints of the existing paths, the perturbation algorithm's expensive root-solve method can fail to generate a valid path, leading to complex code handling of these failure cases, as well as leading to unpredictable overall performance.

4. NOVEL PATH PERTURBATION NUMERICAL SOLVER FOR FEYNMAN PATH INTEGRAL RADIATIVE TRANSFER*

To address the limitations of the previous work's numerical solver, I propose a path-perturbation numerical solver which utilizes a minimal, efficient poly-line path representation that allows for efficient storage of paths. This was published in the Annals of Nuclear Energy and was presented at PHYTRA 5 [36]. Additionally, the proposed path-perturbation solver utilizes a novel and efficient path-perturbation algorithm with better properties than the root-solver perturbation algorithm, such as an efficient mapping to GPU hardware. Note that while I later present a novel path generation numerical solver, I present my path perturbation numerical solver here as the principles still hold.

As discussed, the existing Feynman Path Integral formulation of radiative transfer is a powerful representation of the problem, able to capture essentially all relevant scattering within a volume. While a prior numerical solution for the FPI was published, providing an approach to calculating the FPI, the solver remains incredibly limited. As discussed in the previous section, prior work presented a Monte Carlo numerical solution of the FPI based on perturbing discrete Frenet-Serret representations of paths through a virtual space [13]. While this work was able to calculate an FPI to convergence and perform simple verification experiments, the method was incredibly slow, on the order of CPU months. Thus, numerically calculating the FPI has remained too expensive for use for any real problem, with any computations limited to single transport kernel calculations, or limited virtual environments due to the significant computation requirement.

*Reprinted with permission from "Path integral radiative transfer via polyline representation allowing GPU implementation" by Brennen Taylor, John Keyser, Jerry Tessendorf, 2022. Annals of Nuclear Energy, 173, 109098, Copyright 2022 by Elsevier.

4.1 Proposed Curve Representation

For my novel path-perturbation numerical solver, I proposed the following representation of a discrete path $\mathbb{P}$ with M segments $\mathbb{S}$, visualized in Figure 4.1. In this representation, each path $\mathbb{P}$ is modeled as a set of $M+1$ points, $\{\vec{\boldsymbol{x}}_0, ..., \vec{\boldsymbol{x}}_M\}$, where each pair of points, $\vec{\boldsymbol{x}}_i$ to $\vec{\boldsymbol{x}}_{i+1}$, is defined to be the segment $\mathbb{S}_i$ of length $\Delta s = S/M$. With this definition, as the limit $M \to \infty$ with $M\Delta s = S$, this representation converges to a continuous representation of the path $\mathbb{P}$, with up to $M-1$ scattering events represented by the path integral.

In this representation, for a path $\mathbb{P}$ to be considered valid, i.e. meeting the boundary constraints $\mathbb{R}$, the following must hold.

1. $\vec{\boldsymbol{x}}_0 = \vec{\boldsymbol{x}}_S$
2. $\vec{\boldsymbol{x}}_1 = \vec{\boldsymbol{x}}_0 + \Delta s \hat{\boldsymbol{\omega}}_S$
3. $\vec{\boldsymbol{x}}_M = \vec{\boldsymbol{x}}_R$
4. $\vec{\boldsymbol{x}}_{M-1} = \vec{\boldsymbol{x}}_M - \Delta s \hat{\boldsymbol{\omega}}_R$

With the proposed representation of a path, all information required for the weighting equation can be trivially reconstructed. For a curve of M segments, $M-1$ curvature values are generated, representing the scattering event at the end of each segment. Each segment's curvature value κ_i is reconstructed as follows; first, M tangents, $\{\hat{\boldsymbol{T}}_0, ..., \hat{\boldsymbol{T}}_{M-1}\}$ are constructed using

$$\hat{\boldsymbol{T}}_i = \frac{\vec{\boldsymbol{x}}_{i+1} - \vec{\boldsymbol{x}}_i}{|\vec{\boldsymbol{x}}_{i+1} - \vec{\boldsymbol{x}}_i|} \tag{4.1}$$

After reconstructing the tangents of each segment, the $M-1$ curvature values $\{\kappa_0, ..., \kappa_{M-2}\}$ can be reconstructed using the standard definition of discrete curvature [35]

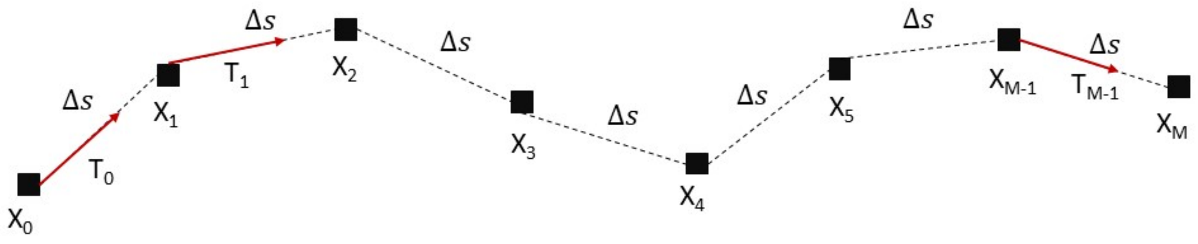


Figure 4.1: Discrete, polyline representation of a curve. In this example, I have M segments, and thus have $M + 1$ stored positions, with $\vec{x}_0$, $\vec{x}_1$, $\vec{x}_{M-1}$, and $\vec{x}_M$ constrained by the boundary conditions $\mathbb{R}$. Each point is located exactly Δs away from its neighbor. Each segment tangent $\hat{T}_i$ (red) is $\vec{x}_{i+1} - \vec{x}_i$, with $\hat{T}_0$, $\hat{T}_{M-1}$ constrained by $\mathbb{R}$.

$$\kappa_i = \left| \frac{\hat{\boldsymbol{T}}_{i+1} - \hat{\boldsymbol{T}}_i}{\Delta s} \right| \tag{4.2}$$

The calculated curvature values are then fed into the segment weighting equation (3.18), and a path weight is calculated via (3.17).

I utilize this representation for the following presented numerical solvers in the remainder of the dissertation. In summary, for a M segment path, I store $M + 1$ points along the path, with the first and last segment points constrained to enforce the boundary conditions $\mathbb{R}$.

4.2 Simplified Geometric Perturbation Numerical Solver

Using the previously discussed poly-line representation, I devised a simple, yet effective and efficient method for perturbing an existing path $\mathbb{P}$ into a new path $\mathbb{P}'$. Unlike the prior work root-solve approach, my perturbation algorithm always guarantees a novel path that meets the constraints required by the boundary conditions.

The method works as follows. First, two points, $\vec{\boldsymbol{x}}_l$ and $\vec{\boldsymbol{x}}_r$, are selected from the curve poly-line uniformly at random. These positions are selected such that they meet the constraints $1 \leq l < (M-1), l < r \leq (M)$ to ensure that $\vec{\boldsymbol{x}}_l$ exists earlier in the path than $\vec{\boldsymbol{x}}_r$ and that there is at least one segment distance Δs separating the two selected points. Secondly, the point selection is constrained such that $\mathbb{S}_0$ and $\mathbb{S}_{M-1}$ are never perturbed, as they must remain fixed to maintain $\mathbb{R}$. Selecting $\vec{\boldsymbol{x}}_l, \vec{\boldsymbol{x}}_r$ divides the discrete poly-line into three sections, each acting as rigid collections of segments that do not change position or rotation with respect to each other throughout the perturbation. The three rigid sections are defined as $RB_0 = \{\mathbb{S}_0, ..., \mathbb{S}_{l-1}\}$, $RB_1 = \{\mathbb{S}_l, ..., \mathbb{S}_{r-1}\}$, $RB_2 = \{\mathbb{S}_r, ..., \mathbb{S}_{M-1}\}$. Throughout a path perturbation, RB_0 and RB_2 are locked, allowing only the positions that makeup RB_1 to rotate. A rotation axis $\vec{\boldsymbol{x}}_r - \vec{\boldsymbol{x}}_l$ is defined and a random angle θ is selected uniformly at random from the range $[0, 2\pi]$. A rotation matrix R_θ around the axis $\vec{\boldsymbol{x}}_r - \vec{\boldsymbol{x}}_l$ is then constructed, and used to rotate each point in RB_1, $\vec{\boldsymbol{x}}_i | l < i < r$, to a newly rotated position $\vec{\boldsymbol{x}}'_i$, shown in Equation 4.3.

$$\vec{\boldsymbol{x}}'_i = \boldsymbol{R_\theta} \left(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_l \right) + \vec{\boldsymbol{x}}_l \tag{4.3}$$

Finally, a newly generated path $\mathbb{P}'$ is constructed by concatenating RB_0, RB'_1, RB_2, such that it consists of the points of $\{\vec{\boldsymbol{x}}_0, ..., \vec{\boldsymbol{x}}_l, \vec{\boldsymbol{x}}'_{l+1}, ..., \vec{\boldsymbol{x}}'_{r-1}, \vec{\boldsymbol{x}}_r, ..., \vec{\boldsymbol{x}}_M\}$. One can easily see that for all segments $\mathbb{S}'_i$ in $\mathbb{P}'$, $|\mathbb{S}'_i| = |\mathbb{S}_i| = \Delta s$, as well as $\{\vec{\boldsymbol{x}}'_0, \vec{\boldsymbol{x}}'_1, \vec{\boldsymbol{x}}'_{M-1}, \vec{\boldsymbol{x}}'_M\} = \{\vec{\boldsymbol{x}}_0, \vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_{M-1}, \vec{\boldsymbol{x}}_M\}$ and $\{\hat{\boldsymbol{T}}_0, \hat{\boldsymbol{T}}_{M-1}\} = \{\hat{\boldsymbol{T}}_0, \hat{\boldsymbol{T}}_M\}$.

By selecting both the left and right pivot points uniformly at random, as well as rotation angle θ uniformly at random, no bias is inserted into the perturbation method. By performing a direct, geometric perturbation of the previous path $\mathbb{P}$ into a new path $\mathbb{P}'$, this novel perturbation method is guaranteed to generate a new path, has a fixed number of operations, and is extremely fast. In addition, each perturbation is guaranteed to meet all boundary constraints of the path integral, always generating valid paths from valid paths.

In comparison, the previous method's root-solve-based path perturbation method was expensive and failed to solve occasionally while introducing a bounded error to each perturbation event. Compared to the previous numerical solver, the presented perturbation method provides much stronger performance guarantees. A visualization of 10^5 path perturbations can be seen from the side in Figure 4.2.

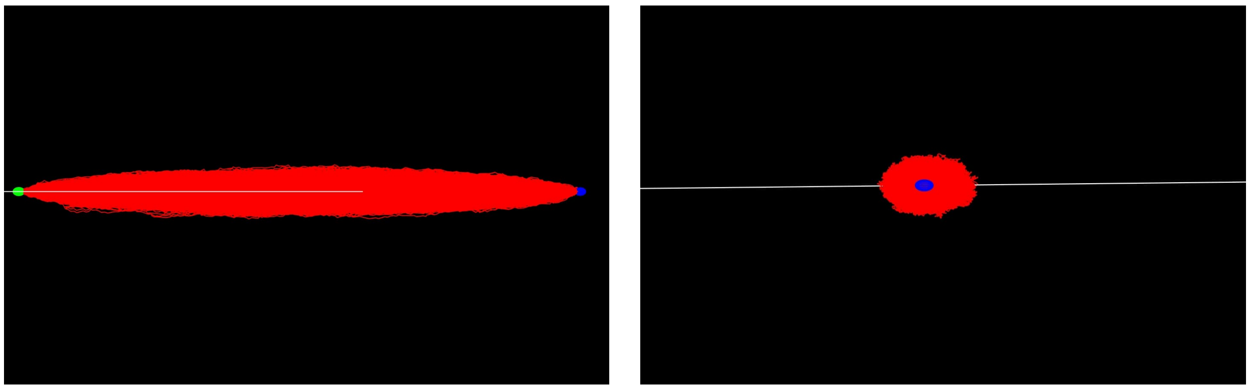


Figure 4.2: Visualization of 10^5 path perturbations (red) in order of generation. One can clearly see how the perturbation method samples the region of space between the starting point $\vec{x}_S$ (green), and the receiver point $\vec{x}_R$ (blue). These points are 10 units apart, with the paths being 12 units in arclength S. The visualized grid (white) is centered on the origin and visualizes the $\{\vec{x}, \vec{z}\}$ plane. Note how the paths are constrained within the expected ellipsoid boundary.

4.2.1 GPU Implementation

One major benefit of the poly-line path representation and simple geometric perturbation algorithm is how it allows for a performant GPU implementation. Unlike the previous perturbation method, the geometric perturb method proposed in this paper has a bounded number of operations per path generated and a consistent execution sequence that minimizes the divergence of threads. Compared to the root-solve based perturb method, which has an unpredictable number of execution paths depending on the random curvature modification, the presented method is a better candidate for the GPU.

Additionally, the presented method is guaranteed to provide a valid path for each path perturbation; i.e. for any path passed in, any perturbation performed on the path maintains the end constraints of the path integral. This simplifies the GPU implementation significantly by allowing the implementation to assume that a path generation will never fail, and thus the number of generated paths desired by a single thread is guaranteed without requiring extra looping or a condition check to ensure that the correct number of paths have been generated. In contrast, the root-solve perturbation method fails 20% of the time [14], requiring a GPU implementation of the root-solver perturbation algorithm to tolerate these failures, introducing branching and looping complexity to the implementation, again increasing thread divergence.

My GPU implementation offloads all path perturbation and a significant portion of the path weighting to the GPU, see Figure 4.3. Experiment execution begins on the CPU, where the CUDA API is used to allocate space per device hardware thread for a curve and copy the initial seed curve's poly-line to each device thread's allocated space. Each thread's curve storage is seeded with the same Bezier-based initial curve, generated via the same methodology as the previous numerical solver[14]. Then, given a problem of N total experiment paths, the problem is first subdivided into K groups of 10^6 paths, per the combined path weight discussed in the next few paragraphs. Given a warp size of W, each warp is assigned to generate K/G combined path weights. With this construction, within a warp of size W, each thread is responsible for generating $10^6/W$ paths and combining their weights into a thread-specific combined path weight. Finally, once all paths have been generated, the K combined path weights are copied back to the CPU, and execution switches back to the CPU. The path weights are then extracted from the combined path weights and summed into the final FPI weight.

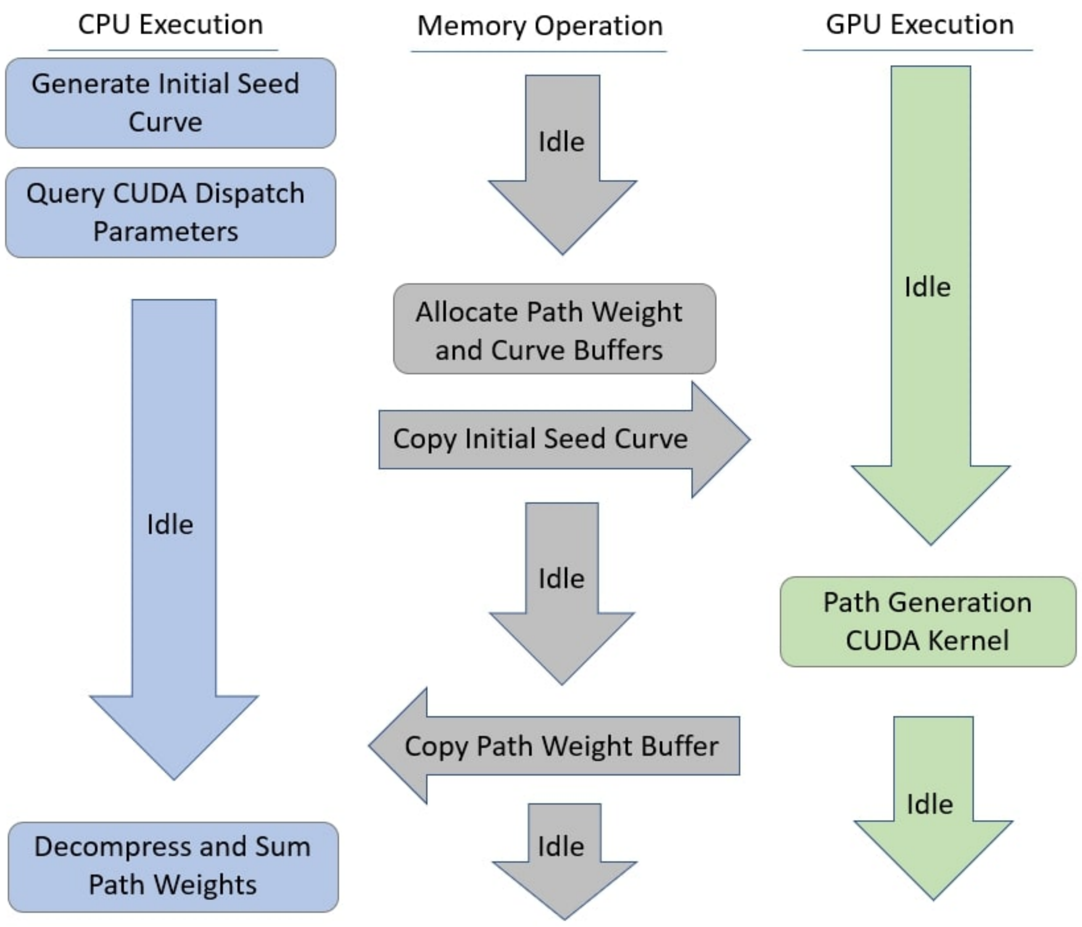

CPU Execution
Memory Operation
GPU Execution
Generate Initial Seed Curve
Query CUDA Dispatch Parameters
Idle
Allocate Path Weight and Curve Buffers
Copy Initial Seed Curve
Idle
Path Generation CUDA Kernel
Copy Path Weight Buffer
Idle
Decompress and Sum Path Weights

Figure 4.3: Flow diagram describing the following GPU implementation steps: 1. An initial seed curve is generated on the CPU. 2. The CUDA API is queried for the optimal dispatch parameters for the perturb GPU kernel. 3. Memory is allocated on the GPU for storing combined path weights and paths scratch space per thread. 4. The initial curve is copied to the GPU per thread. 5. The GPU kernels are dispatched and all desired paths and combined path weights are generated. 6. Combined path weights are transferred back to the CPU. 7. Combined path Weights are decompressed and summed. Refer to Table 4.3 for a more detailed breakdown of the runtime of each phase.

In my implementation, I aim to minimize the amount of data transfer required between the host (CPU) and device (GPU), particularly concerning scaling with the number of desired paths for the experiment. During the initial experiment setup, a warp size W and grid size G is selected, bounding the maximum number of unique threads executed on the device. Each of these threads is allocated unique space for storing the positions, tangents, and curvatures of the curve it is perturbing. As each perturbation is guaranteed to generate a valid curve, each thread simply overwrites its previous thread with each perturbation event, thus minimizing memory usage. The experiment setup requires the copy of the initial seed curve from host to device memory per thread and thus is bounded only by the number of threads dispatched rather than the overall number of paths requested for the experiment. Additionally, each thread's per thread combined path weight is stored in shared memory. As this memory is accessed often (per path generated) and small in size (only four doubles), it is an ideal candidate for shared memory and extremely performant.

Unlike the setup stage, the transfer of data back from the device to the host scales with the number of paths in the experiment, as ultimately, the data transferred back must contain the weight of each path generated in the experiment. To minimize the amount of data transferred back, I devise a compression method to store 10^6 paths in a combined path weight, represented by three doubles and a single integer. In this scheme, the maximum path weight stored in each combined path weight batch is scaled so it is the maximum possible double. All other path weights stored in the same combined path weight are likewise scaled such that their relative weight to the maximum path weight is maintained. The combined total of the path weights in the batch is stored in one double, while the offset required to map the max path weight to the max double value, the current max path weight in the combined path weight, and the number of values stored in the combined weight are stored in the remaining variables. Once a combined weight value has been filled with up to 10^6 path weights, the combined running total and offset can be transferred back to the CPU and decompressed for the final weight combination. Each compressed weight is converted to an arbitrary precision floating point value, a $BigFloat$, and all are summed into the final path weight representing all N paths in the experiment.

Table 4.1: CPU Implementation Performance

Path Count	Run 1(ms)	Run 2(ms)	Run 3(ms)	Run 4(ms)	Run 5(ms)	Average(min)
10^6	13517	13622	13483	13505	13491	0.23
10^7	78531	78784	78708	78714	78633	1.31
10^8	731451	734465	729769	733256	730593	12.20
10^9	7268669	7323906	7353347	7254203	7360236	121.88

Table 4.2: GPU Implementation Performance

Path Count	Run 1(ms)	Run 2(ms)	Run 3(ms)	Run 4(ms)	Run 5(ms)	Average(min)
10^6	14150	14154	14171	14153	14210	0.24
10^7	26586	26615	26618	26603	26598	0.44
10^8	196835	196718	196724	196672	196783	3.28
10^9	1884777	1883359	1883230	1883345	1885421	31.40

4.2.2 CPU and GPU Performance Comparison

To compare the CPU and GPU implementations, I generated a specified number of paths ranging from 10^6 to 10^9 for each implementation, comparing their run times. These experiments were performed on an Intel i7-8700 with 12 logical processors and 16GB of RAM as the host, using an Nvidia Titan V as the GPU device. The CUDA dispatch parameters were tuned for the CUDA 7.2 architecture, with a dispatch warp size of 64 and a grid size of 64.

Tables 4.1 and 4.2 show that both the CPU and GPU implementations scale linearly with the number of paths in the experiment. As both the CPU and GPU implementations' setup phases take constant time independent of the number of paths, scaling only with the parameters of the system,

Table 4.3: GPU Implementation Breakdown

Path Count	Avg. Run Time(min)	% Time(Setup)	% Time(Perturb)	% Time(Weighting)
10^6	0.24	46.98%	52.99%	0.03%
10^7	0.44	24.94%	74.95%	0.11%
10^8	3.28	3.38%	96.48%	0.14%
10^9	31.40	0.38%	99.48%	0.14%

a small number of paths allows for the CPU implementation to potentially perform better than the GPU as the CPU setup time is slightly cheaper than that of the GPU. Then as the number of paths increases, the perturbation phase takes a larger percentage of the full computation, with the GPU implementation outperforming the CPU implementation by a constant factor of 4x.

Focusing specifically on the GPU implementation, Table 4.3 shows how the implementation is computationally bound by the compute phase due in part to the techniques discussed previously regarding a reduced memory transfer between the host and device.

Additionally, it is important to note the performance difference between my CPU and GPU implementations to that of the previous method. While data is limited for the previous method, I reference the beam spread experiment data presented in Kilgo's dissertation to compare time metrics [28].

For this comparison, I discuss metrics in terms of the throughput of path generation. Unfortunately, this is not exactly a one-to-one comparison, as the metrics reported by Kilgo are performed over many iterations of separate FPI-RT invocations of 1000 perturbed paths and initial curve generations. In contrast, my method is a single execution of an FPI-RT with a significantly higher path count. However, as discussed in the next section, I can easily construct an initial curve at the same cost as generating a single path sample. Thus, I believe that a direct comparison of the throughput is fair.

For the previous method, $4x10^5$ batches of 1000 perturbations of paths take 1.15 years, leading to a generation of $4x10^8$ total paths in 1.15 years. In contrast, my method on the CPU generates 10^8 paths in 12.2 minutes, or $2.31x10^{-5}$ years, resulting in a speed increase of $12394x$ increased throughput, or four orders of magnitude. Likewise, my GPU implementation generates 10^8 paths in 3.21 minutes, or $6.24e^{-6}$ years, resulting in a speed increase of $46100x$ increased throughput, again four orders of magnitude. Thus, my method demonstrates very clear performance wins over the previous method.

4.2.3 Failure of Geometric Perturbation

While my published numerical solver significantly improved over the existing numerical solvers of the radiative transfer path integral, several issues remained. Despite my previous validation experiments of the path perturbation numerical solver signaling overall convergence, further discussed in Appendix C, I found that the solver still struggled with convergence in many environments. In contrast to expected Monte Carlo behavior, I found that the numerical solution began to diverge as larger numbers of paths were included, with the final converged weight dropping off significantly and steadily as the number of paths increased.

The main symptom of the problem is the consistent trend of a sequence of perturbed paths, ultimately resulting in a sequence of low-weighted paths. I observe that once a sequence of generated paths has reached this low-weighted state, the path perturbation method is unlikely to explore any paths outside a very small subspace of possible paths. I deem this behavior ’coiling,’ as visualizing the generated sequence of low-weighted paths shows a grouping of very similar, largely overlapping paths in which the full lengths of the paths are tightly woven into a small region of overall space.

My understanding is that my geometric perturbation and the path perturbation solvers are not without bias, introduced by the parameters of the path perturbation impacting multiple degrees of freedom of the model at a time. During a path perturbation, while I select a starting and ending segment uniformly at random and a rotation angle uniformly at random, the resulting rotations do not generate novel, valid path samples which are significantly different across valid path space. Instead, the bias introduced by mapping the selected uniform random parameters for the geometric perturbation causes a sequence of generated paths to, despite rotation, coil into a small region of valid path space. While escaping this coiled state is possible, the required set of rotation events is significantly less likely than a sequence of rotations that maintain the coiled state.

I believe this inability of the geometric perturbation algorithm to modify the path space degrees of freedom (the angles discussed in the next section) uniquely ultimately results in the sequence of paths becoming stuck in a 'coiled' state and producing only low-weighted paths. Further exploration of the exact means for this may be worthwhile. However, I explore methods for directly sampling path space, as described in the next section.

5. NOVEL PATH GENERATION NUMERICAL SOLVER FOR FEYNMAN PATH INTEGRAL RADIATIVE TRANSFER

This chapter begins by taking a step back and presenting a different approach to viewing paths from the perspective of the independent angles connecting neighboring segments. With this new perspective, I present a novel method for generating unique, valid paths, which are guaranteed to meet specified boundary conditions. With this new path generator, I then present a path-generation numerical solver for computing the FPI, which I find to outperform the path-perturbation approach presented in the previous chapter significantly.

5.1 Path Geometry

In the remaining sections of this work, it will often be helpful to discuss paths in terms of exploring the total valid path space possible given a set of boundary conditions $\mathbb{R}$ and the independent degrees of freedom used to model such paths. Specifically, I discuss paths in terms of manipulating the spherical coordinate angles of each segment, $\mathbb{S}_{i+1}$ with respect to its previous neighboring segment $\mathbb{S}_i$, as shown in Figure 5.1. I continue with a discussion of path construction for paths with small numbers of segments.

5.1.1 Fully Constrained Paths, Fewer Than Four Segments

Starting with paths of segment count one, I find that these paths are valid only for an extremely constrained set of boundary conditions. A valid path with one segment can only exist when the boundary conditions are set such that $|\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S| = S$ and $\hat{\boldsymbol{\omega}}_R = \hat{\boldsymbol{\omega}}_S = \left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S\right) / (|\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S|)$, as seen in Figure 5.2a.

For paths with segment count two, again only a single valid path, if any, exists per boundary condition. A valid path exists only for environments constructed such that $\vec{\boldsymbol{x}}_S + \Delta s \hat{\boldsymbol{\omega}}_S = \vec{\boldsymbol{x}}_R - \Delta s \hat{\boldsymbol{\omega}}_R$. This construction is shown in Figure 5.2b.

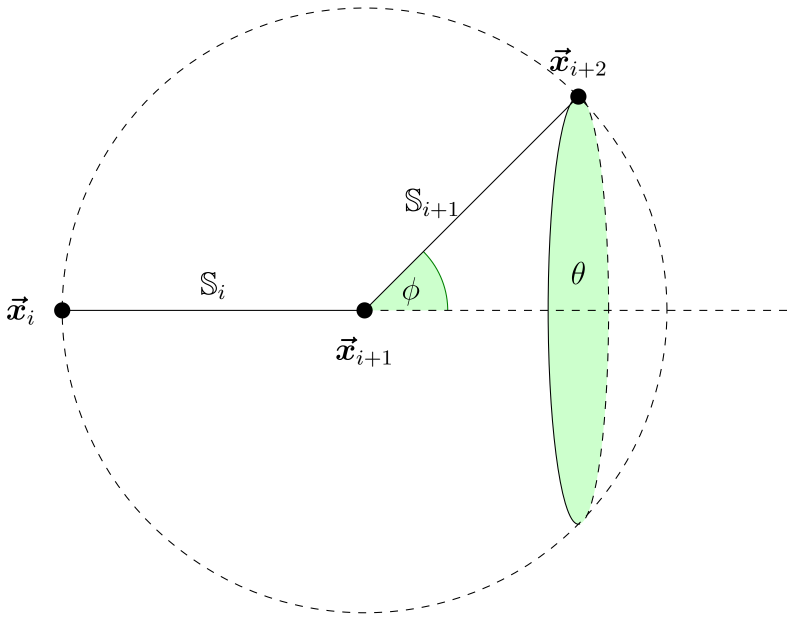


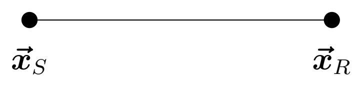
Figure 5.1: With two neighboring segments, I can define segment $\mathbb{S}_{i+1}$ in terms of spherical coordinates off of $\mathbb{S}_i$. Starting at position $\vec{\boldsymbol{x}}_{i+1}$, I place $\mathbb{S}_{i+1}$ an angle ϕ off of the axis formed by $\hat{\boldsymbol{T}}_i$. I then rotate $\mathbb{S}_{i+1}$ an angle θ around this axis.

Lastly, I look at paths of segment count three. With three-segment paths, paths remain quite constrained, with path construction shown in Figure 5.2c. While the number of environment configurations supported continues to increase, for a given boundary constraint to have a valid solution, $|(\vec{\boldsymbol{x}}_R - \Delta s\hat{\boldsymbol{\omega}}_R) - (\vec{\boldsymbol{x}}_S + \Delta s\hat{\boldsymbol{\omega}}_S)| = \Delta s$. Thus, only a single valid path can exist for a given boundary constraint.

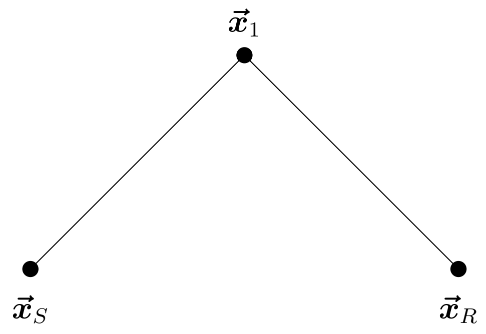
I note that, while important to specify for a bottom-up construction approach, paths of segment count $M \leq 3$ do not make sense for the application of a numerical solver as there is no valid path space to integrate over; only one valid path exists per set of boundary conditions.

5.1.2 Paths With Free Angles, at Least Four Segments

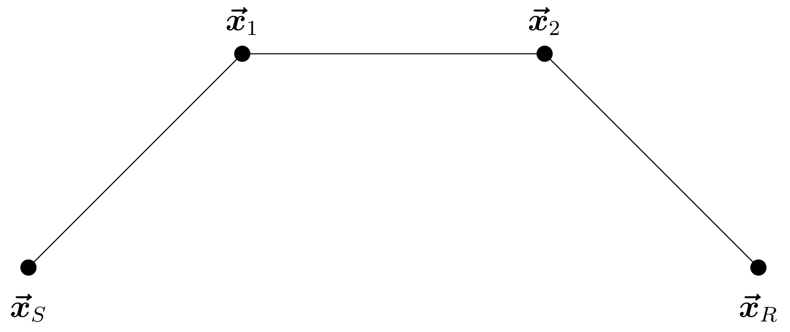
Now, I shift our attention to those paths constructed with more than three segments. I find that for all paths with at least four segments, the size of valid path space increases such that there is more than one valid path, requiring integration over valid path space to compute $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$. Thus, paths with at least four segments are perfect candidates for solving using my presented FPI



(a) One segment path.



(b) Two segment path.



(c) Three segment path.

Figure 5.2: All segments are defined solely by the boundary conditions. In all cases, only a single path meets the qualifications for being valid.

solver.

5.1.2.1 Four Segment Paths

I begin with a discussion of four segment paths which, as mentioned, allow for multiple valid paths for a given boundary constraint configuration. So long as $|(\vec{x}_R - \Delta s\hat{\omega}_R) - (\vec{x}_S + \Delta s\hat{\omega}_S)| \leq 2\Delta s$, an infinite number of valid paths exist meeting the boundary conditions. With the introduction of a fourth segment, I have introduced the first angle degree of freedom, θ_0, to the constructed paths. As can be seen in Figure 5.3, for a given boundary constraint, all valid paths are configurations of the two center segments' intersection point, $\vec{x}_2$ rotated around a center axis defined by $\vec{x}_3 = \vec{x}_R - \Delta s\hat{\omega}_R$ and $\vec{x}_1 = \vec{x}_S + \Delta s\hat{\omega}_S$. I place point $\vec{x}_2$ by rotating the first free segment, $\mathbb{S}_1$, around axis $\vec{x}_1\vec{x}_3$ by θ_0 which can take values in the range $[0, 2\pi]$. Additionally, I note the introduction of another angle, ϕ_0, defining the angle of $\mathbb{S}_1$ off of axis $\vec{x}_1\vec{x}_3$. However, as $\mathbb{S}_1$ and $\mathbb{S}_2$ have fixed

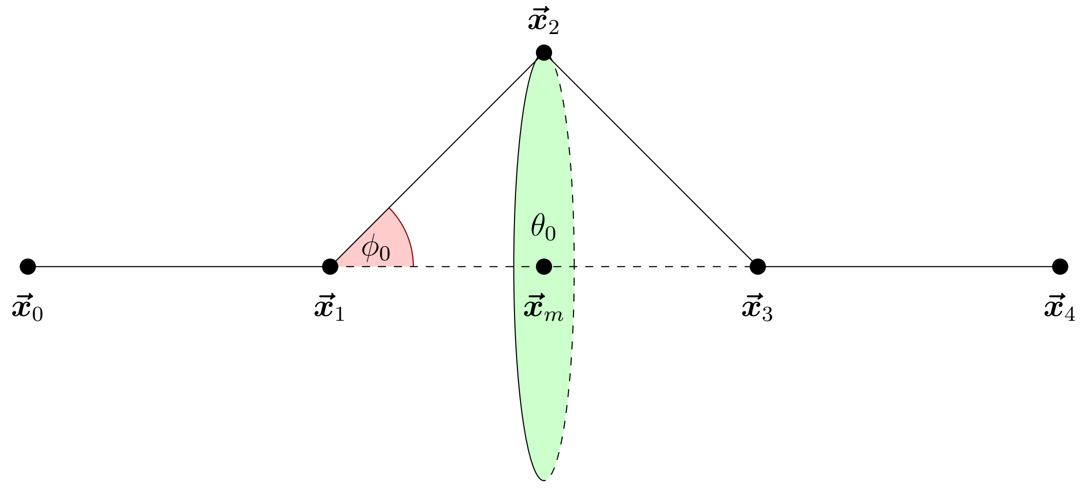


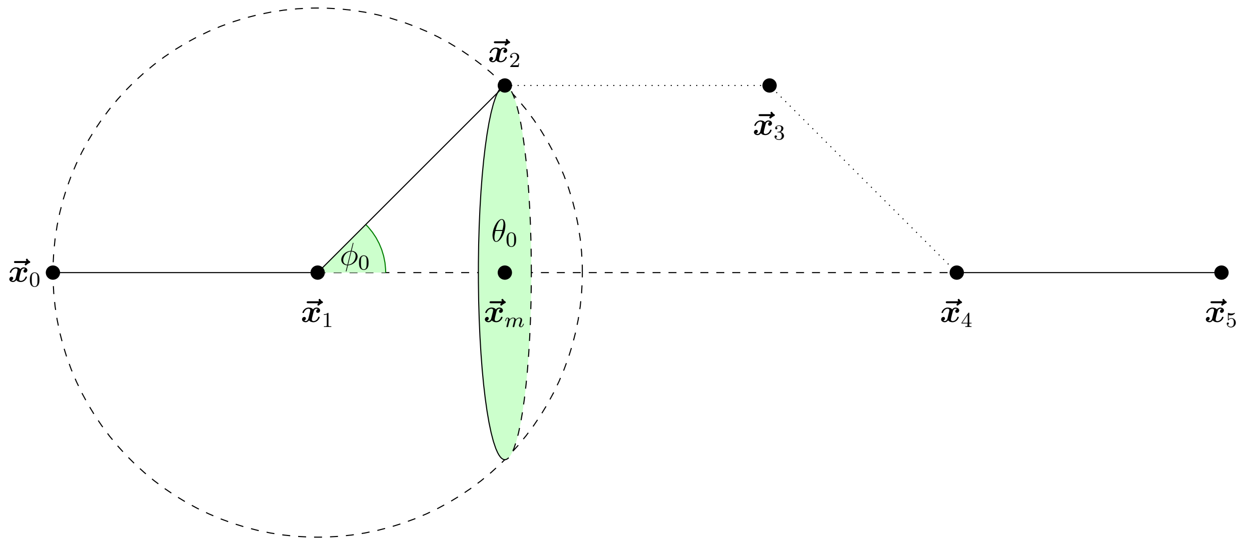
Figure 5.3: With four total segments, the center two segments are free to rotate around the axis defined by $x_1 x_3$ (green) by angle θ_0. However, the angle of segment S_{12} (red) off this axis ϕ_0 is set by the boundary constraints.

lengths encoded by the path construction delta functions, this angle ϕ_0 is enforced by the boundary conditions and path geometry, and thus is constrained to a single value. In summary, I find that a four-segment path can be paramaterized as $\mathbb{P}_{M=4}(\mathbb{R}, \theta_0)$.

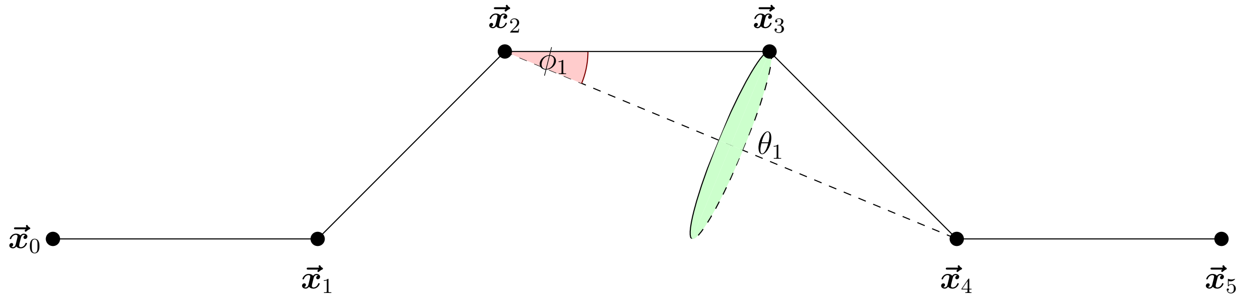
5.1.2.2 Five Segment Paths

Next, consider paths with an additional fifth segment. I construct these paths by first placing segment $\mathbb{S}_1$, the first movable segment, around a sphere of radius Δs focused on $\vec{x}_1$, as shown in Figure 5.4a. This segment placement is defined in spherical coordinates by two angles, ϕ_0 from $[0, \pi]$ and θ_0 from $[0, 2\pi]$. However, selections of ϕ_0 and θ_0 remain valid so long as the remaining distance $|\vec{x}_4 - \vec{x}_2| \geq 2\Delta s$ after placing $\mathbb{S}_1$ is maintained.

After placing $\mathbb{S}_1$, the two remaining free segments, $\mathbb{S}_2, \mathbb{S}_3$ are placed similarly to the four-segment paths above. The rotation of $\mathbb{S}_2$ around $\vec{x}_2\vec{x}_4$, where $\vec{x}_2$ is set by the placement of $\mathbb{S}_1$, is defined by θ_1, which is free to rotate between $[0, 2\pi]$, shown in Figure 5.4b. Additionally, $\mathbb{S}_2$ is placed an angle ϕ_1 off of the axis of rotation defined by $\vec{x}_2\vec{x}_4$. Again, as $\mathbb{S}_2$ and $\mathbb{S}_3$ have fixed lengths encoded



(a) With five segments, there are three center segments that are free to move: $\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3$. Segment $\mathbb{S}_1$ is placed first by selecting a valid angle ϕ_0 of the axis $\vec{x}_1\vec{x}_4$ (green). Additionally, the segment is then rotated θ_0 around axis $\vec{x}_1\vec{x}_4$. All configurations of ϕ_0 and θ_0 are valid such that $|\vec{x}_4 - \vec{x}_2| \leq 2\Delta s$.



(b) After setting segment $\mathbb{S}_1$, segments $\mathbb{S}_2$ and $\mathbb{S}_3$ are placed. Like the four-segment case, these two remaining segments are constrained by the boundary conditions such that $\mathbb{S}_2$ is a fixed ϕ_1 off of the axis $\vec{x}_2\vec{x}_4$. However, they are free to rotate θ_1 around axis $\vec{x}_2\vec{x}_4$.

Figure 5.4: Five segment path.

by the path construction delta functions, this angle ϕ_1 is enforced by the boundary conditions and path geometry, and thus is constrained to a single value. With this, I have that a five-segment path can be parameterized as $\mathbb{P}_{M=5}(\mathbb{R}, \theta_0, \phi_0, \theta_1)$. With the introduction of an additional, fifth segment, I have introduced two additional angles compared to the four-segment paths.

5.1.2.3 M Segment Paths

I now define the general case. For a M segment path, $M-2$ segments are unspecified by the boundary conditions and thus can be freely placed. I place each free segment by assigning a ϕ and

θ pair of spherical coordinates defining that segment's rotation around its starting position, except for the last free segment which only is specified by a single θ parameter, as the ϕ parameter is set by the boundary conditions and path geometry. Thus, a path with M segments $\mathbb{P}_M$ is parameterized by $(2M - 5)$ parameters.

In general, path construction allows for ϕ to range from $[0, \pi]$ and θ to range from $[0, 2\pi]$. However in practice, as my numerical solver is focused only on valid paths which enforce boundary constraints, each free parameter will have minimum and maximum value constraints in order to remain valid.

5.2 Valid Path Generation

Next, I present a method of randomly generating valid paths with M segments which are guaranteed to meet specified boundary conditions $\mathbb{R}$. While the approach is limited to specific segment counts, as I will discuss in Section 5.2.3, the overall path-generation approach is very powerful. It allows for the generation of valid paths across the full expanse of valid path space.

Beginning with the definition of two base algorithms that construct paths of two and three free segments, I will then construct a recursive path generation algorithm that can generate paths of even numbers of free segments.

5.2.1 Two Free Segment Path Generation

I start with a description of the approach generating (resolving) two free segments with a fixed start and end position. While this can be used to directly generate paths of two free segments (four segment paths), I will find that the approach can be used to generate the configuration of any neighboring two segments, so long as the starting and ending positions are provided.

Lets define the two segments to be $\mathbb{S}_0$ and $\mathbb{S}_1$ spanning points $\vec{\boldsymbol{x}}_0$, $\vec{\boldsymbol{x}}_1$, and $\vec{\boldsymbol{x}}_2$. When placing the two free segments, $\mathbb{S}_0$ and $\mathbb{S}_1$, I have three possible cases which need resolution.

First, as shown in Figure 5.5, it is possible that there is no valid configuration of the segments. In this case, the distance d between $\vec{\boldsymbol{x}}_1$ and $\vec{\boldsymbol{x}}_3$ is larger than $2\Delta s$, the distance which the two segments can span; i.e., the segments cannot connect, and thus no valid path configuration exists.

In a second case, $\vec{\boldsymbol{x}}_0$ and $\vec{\boldsymbol{x}}_2$ are stacked, as shown in Figure 5.6. In this case, I have that segments $\mathbb{S}_0$ and $\mathbb{S}_1$ will also overlap exactly, as the distance of Δs traversed from $\vec{\boldsymbol{x}}_0$ to $\vec{\boldsymbol{x}}_1$ by $\mathbb{S}_0$ must be then traversed from $\vec{\boldsymbol{x}}_1$ to $\vec{\boldsymbol{x}}_2$ by $\mathbb{S}_1$. However, segment $\mathbb{S}_0$ is free to be placed anywhere on the sphere of radius Δs surrounding $\vec{\boldsymbol{x}}_0$, and thus $\mathbb{S}_0$'s θ can be sampled from the full $[0, 2\pi]$ range, and its phi can be sampled from $[0, \pi]$. I do not sample ϕ directly, but rather sample $cos(\phi)$ from $[-1, 1]$ to allow for unbiased, uniform sampling of segment $\mathbb{S}_0$ configurations.

In the third case, $\mathbb{S}_0$ and $\mathbb{S}_1$ intersect along the circumference of a circle centered exactly along the axis $\vec{\boldsymbol{x}}_0\vec{\boldsymbol{x}}_2$, shown in Figure 5.7. In this case, θ can be uniformly sampled from $[0, 2\pi]$ while I solve directly for ϕ using the following equations.

$$d = |\vec{\boldsymbol{x}}_2 - \vec{\boldsymbol{x}}_1|$$
$$\phi = acos\left(\frac{d/2}{\Delta s}\right)$$

With this, I have defined a procedure for randomly generating configurations of two neighboring free segments.

5.2.2 Three Free Segment Path Generation

Next, I discuss the generation of paths consisting of three neighboring free segments $\mathbb{S}_0, \mathbb{S}_1, \mathbb{S}_2$. Unlike the two-segment case, the resolution of three segments will involve two phases. In the first phase, the leading segment $\mathbb{S}_0$ is placed, after which the last two segments are resolved using the two-segment resolution algorithm for the second phase. Like the two-segment approach, the three-segment resolution has multiple cases which must be handled.

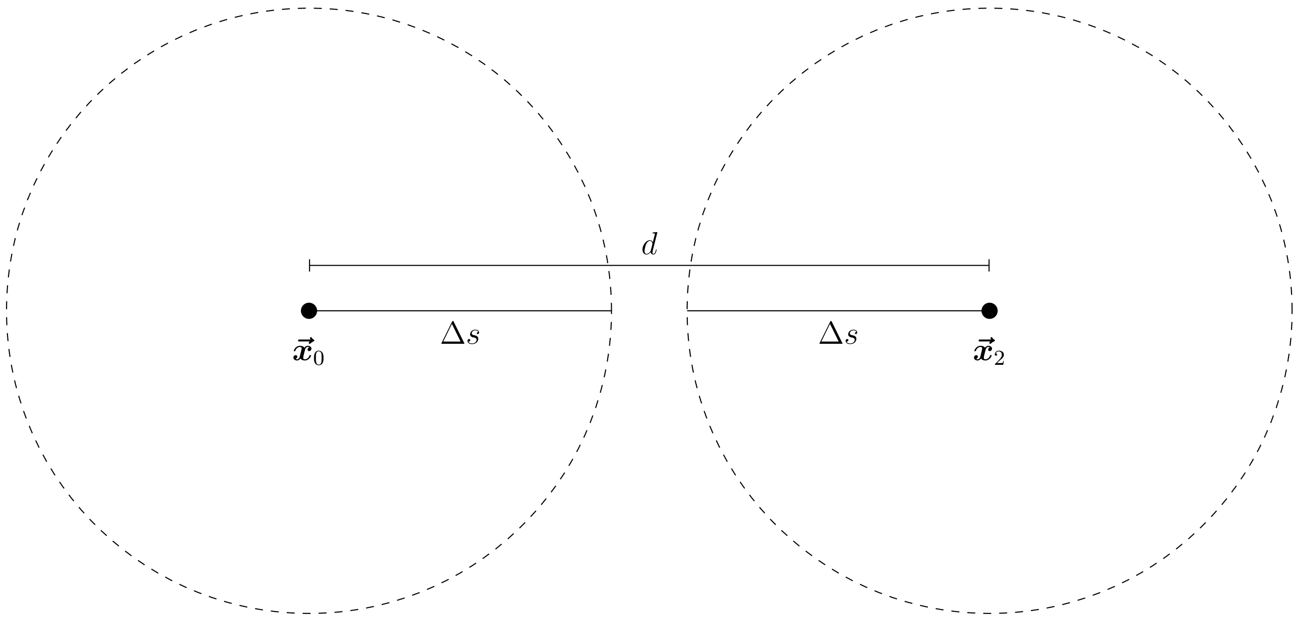


Figure 5.5: With two free segments covering distance $2\Delta s$, but a separation of d with $2\Delta s < d$.

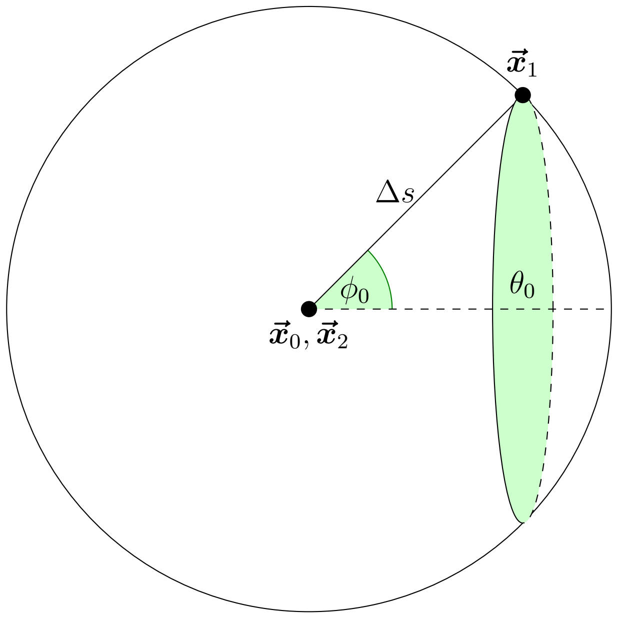


Figure 5.6: When the two points $\vec{x}_0$ and $\vec{x}_2$ overlap, point $\vec{x}_1$ is placed anywhere on the surface of the sphere centered at $\vec{x}_0$ with a radius Δs.

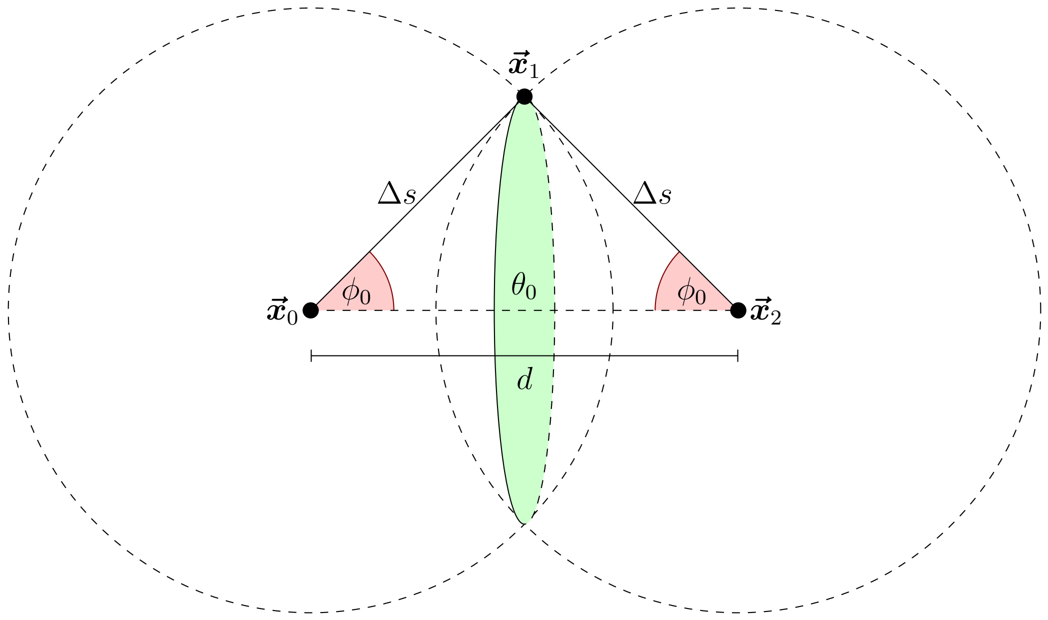


Figure 5.7: With two free segments to place between two points, $\vec{x}_0$ and $\vec{x}_2$, which intersect at $\vec{x}_1$.

The first case again accounts for no valid path configuration. In this case, as seen in Figure 5.8, the distance between points $\vec{x}_0$ and $\vec{x}_3$ is d while the total length of the path spanned by the three free segments $3\Delta s$ is such that $3\Delta s < d$. Thus, no valid path exists in this case, as the segments cannot span the required distance d.

In the second case, points $\vec{x}_0$ and $\vec{x}_3$ are stacked at the same position in space, as shown in Figure 5.9. In this case, the first segment $\mathbb{S}_0$ can simply be placed such that $\vec{x}_1$ is uniformly placed on the sphere of radius Δs centered at $\vec{x}_0$. Then, the remaining two segments are placed using the two-segment resolution. Note, this will always resolve as the distance between $\vec{x}_1$ and $\vec{x}_3$ is Δs, and I have two segments for which to cover the distance.

Lastly, I have a third, more general case for resolving three segments, visualized in Figure 5.10. Similar to the two free-segment generation discussed in the previous section, this case is defined by intersecting the sphere of radius Δs centered at $\vec{x}_1$, representing the extent of the first free segment, with a sphere of radius $2\Delta s$ centered on $\vec{x}_4$, representing the extent of the two remaining

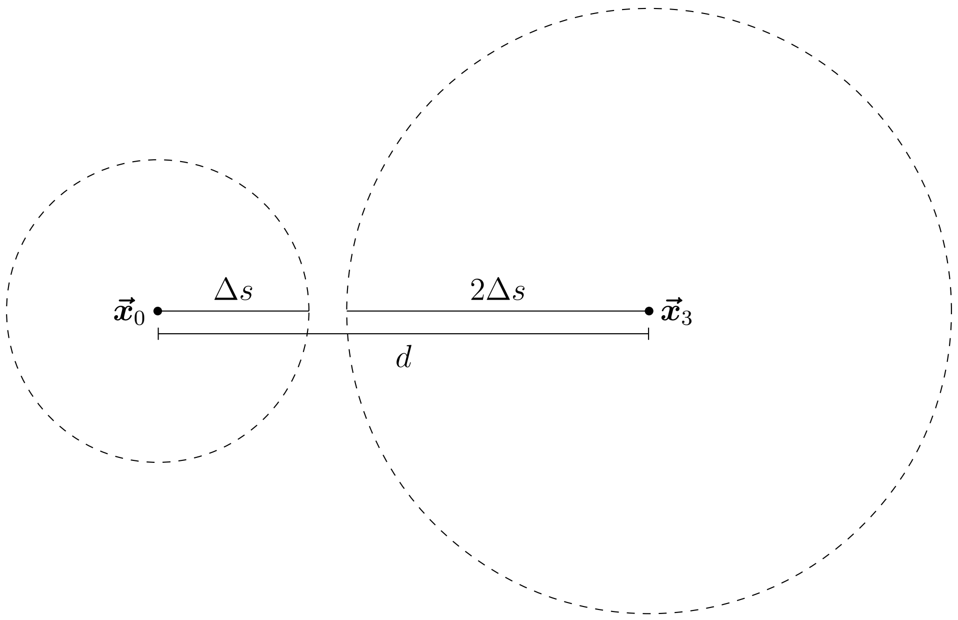


Figure 5.8: With three free segments, but an invalid environment configuration.

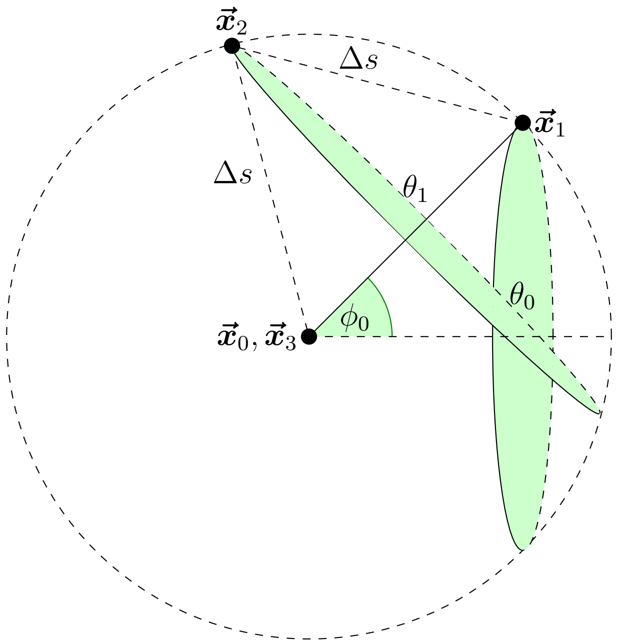


Figure 5.9: When the two points $\vec{x}_0$ and $\vec{x}_3$ overlap, point $\vec{x}_1$ is placed anywhere on the sphere placed centered at $\vec{x}_0$ with a radius Δs.

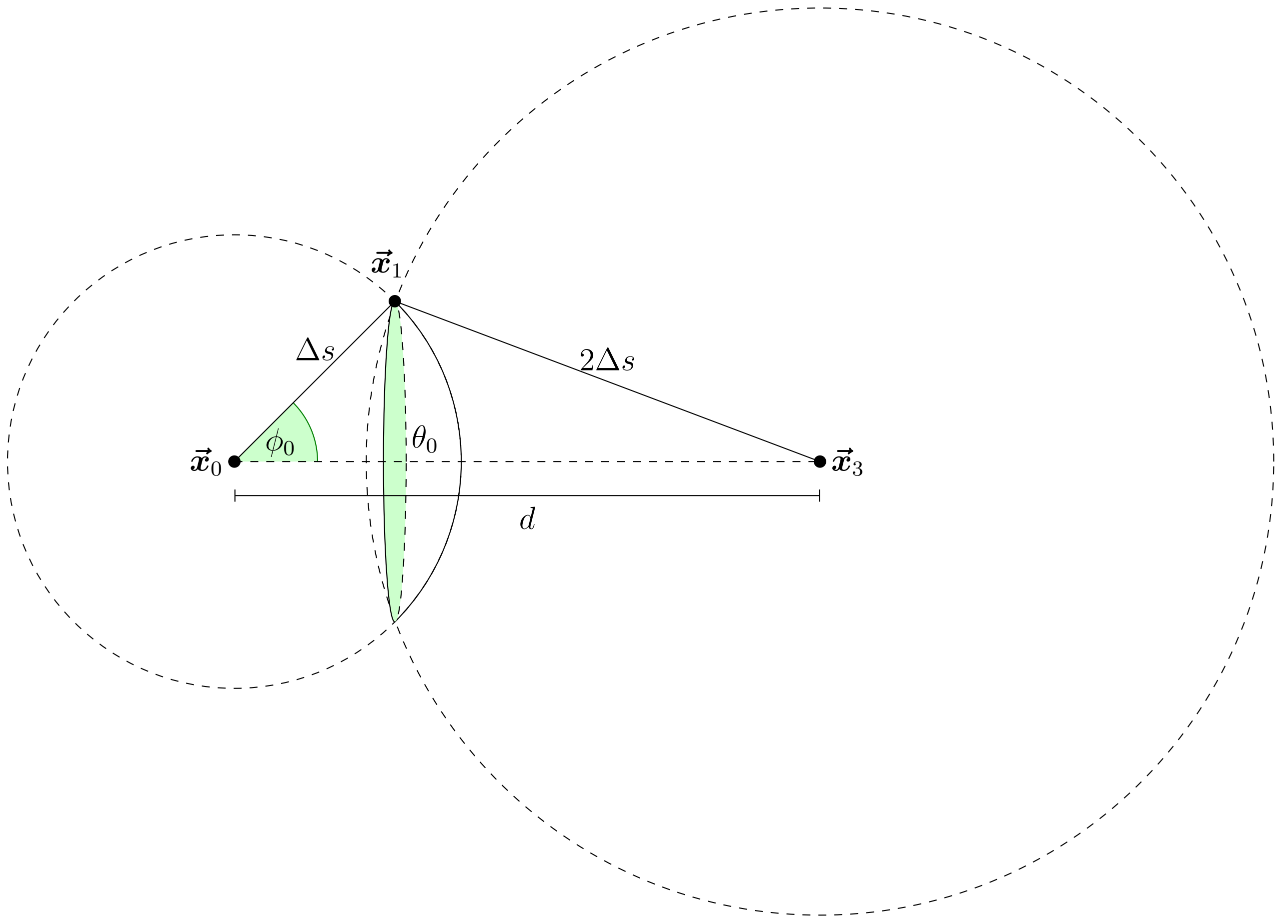


Figure 5.10: With three free segments to place between two points, $\vec{\boldsymbol{x}}_0$ and $\vec{\boldsymbol{x}}_3$, I first intersect sphere extent of the first segment $\mathbb{S}_0$ with the sphere extent of the combined segments $\mathbb{S}_1$ and $\mathbb{S}_2$, intersecting at $\vec{\boldsymbol{x}}_1$. Segments $\mathbb{S}_1$ and $\mathbb{S}_2$ are then resolved using the two-segment resolution.

free segments. This results in two subcases that must be handled.

In the first subcase, if $\mathbb{S}_1$'s sphere is fully contained within the two-segment sphere, $\mathbb{S}_1$ has a full range of motion and can be placed such that $\vec{\boldsymbol{x}}_2$ is uniformly sampled from the surface of $\mathbb{S}_1$'s sphere. Then, after placing $\mathbb{S}_1$, the two remaining segments $\mathbb{S}_2$ and $\mathbb{S}_3$ are resolved using the two-free-segment generation algorithm defined in the previous section.

In the other subcase, shown in Figure 5.10, if the two spheres only partially intersect, $\mathbb{S}_0$'s ϕ angle will be constrained between $[0, \phi_{max}]$. If $\vec{\boldsymbol{x}}_0$ is fully outside the sphere centered at $\vec{\boldsymbol{x}}_3$, I use the following equations to calculate ϕ_{max}. In these equations, h represents the fraction of d from $\vec{\boldsymbol{x}}_0$ to $\vec{\boldsymbol{x}}_3$ at which the intersection circle of the two spheres is centered. a represents the radius of this intersection circle, with the circle placed perpendicular to the axis $\vec{\boldsymbol{x}}_0\vec{\boldsymbol{x}}_3$.

$$h = \frac{1}{2} + \frac{-3\Delta s^2}{2d^2}$$

$$a = sqrt(\Delta s^2 - (hd)^2)$$

$$\phi_{max} = arcsin(a/\Delta s)$$

Instead, if $\vec{\boldsymbol{x}}_0$ is contained within the sphere centered at $\vec{\boldsymbol{x}}_3$, ϕ_{max} is instead calculated as

$$\phi_{max} = \pi - asin(a/\Delta s)$$

With an upper bound for ϕ for segment $\mathbb{S}_0$, I place $\mathbb{S}_0$ by randomly sampling $cos(\phi)$ from $[cos(\phi_{max}), 1]$, and θ_0 from $[0, 2\pi]$. Then, the remaining segments $\mathbb{S}_1$ and $\mathbb{S}_2$ are resolved using the two-segment generation.

With this, I have defined a procedure for randomly generating configurations of three neighboring free segments.

5.2.3 N Free Segment Path Generation

Now, I discuss a general approach for generating paths with N free segments. For a curve with N free segments, I can evenly subdivide the list of free segments into two lists of $\frac{N}{2}$ segments. I then sample the overlapping volume of the two spheres of radius $\frac{N}{2}\Delta s$ centered at $\vec{\boldsymbol{x}}_0$ and $\vec{\boldsymbol{x}}_N$ for the position connecting these two lists of segments, $\vec{\boldsymbol{x}}_{N/2}$. Again, I must handle a few cases.

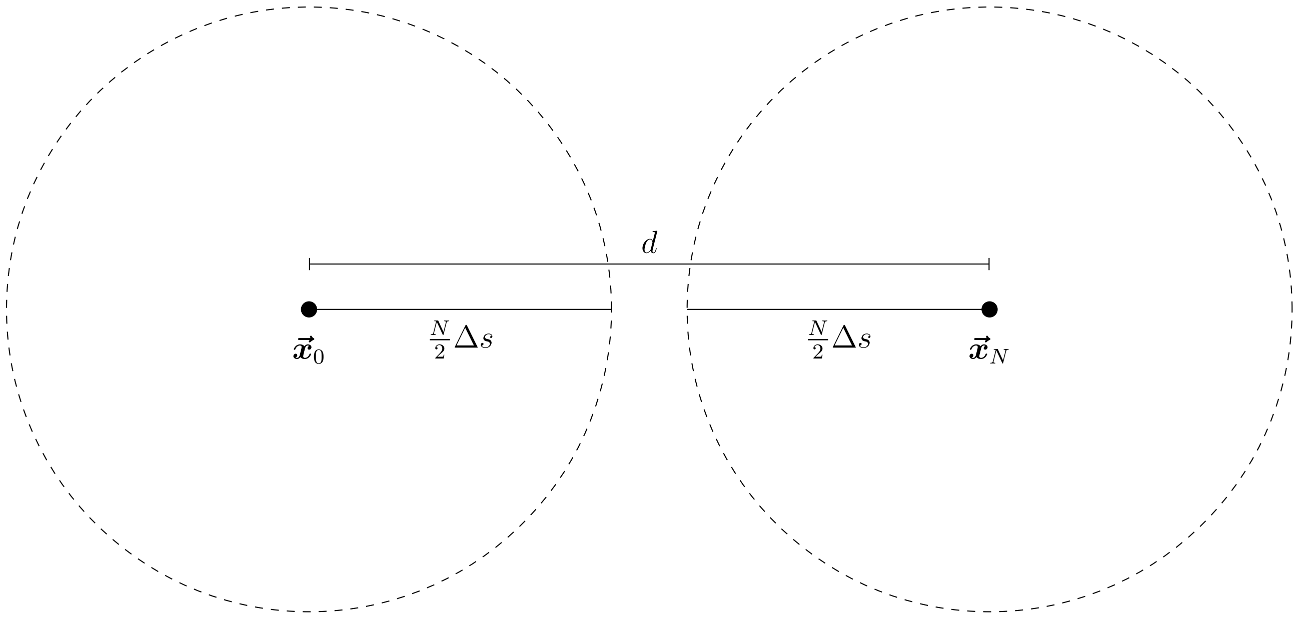


Figure 5.11: When resolving a distance d with N segments, but $N\Delta s < d$, thus no valid path configuration exists.

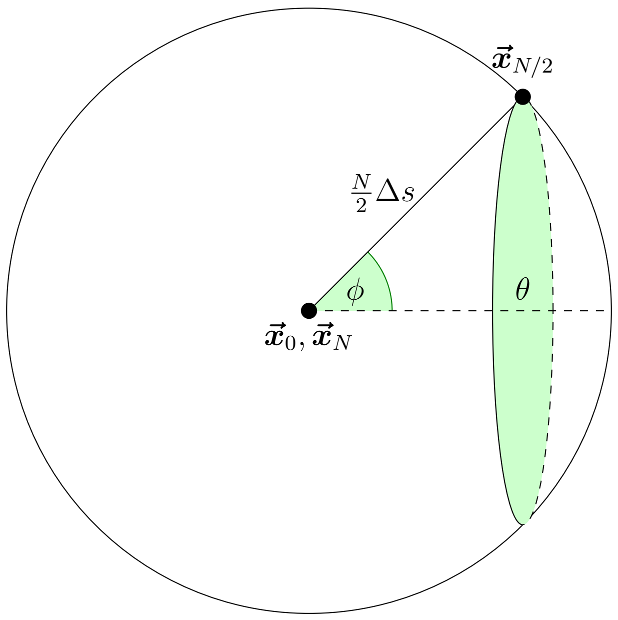
First, I have the case where no valid path exists, shown in Figure 5.11. In this case, the distance d between the two points $\vec{x}_0$ and $\vec{x}_N$ is greater than $N\Delta s$, the distance spanned by the length of the path. Thus no valid path can exist.

In a second case, I have that $\vec{x}_0$ and $\vec{x}_N$ are stacked, and thus the connecting point $\vec{x}_{N/2}$ can be placed anywhere in the sphere of radius $(N/2)\Delta s$ centered at $\vec{x}_0$, shown in Figure 5.12. In this case, I generate three parameters for placing the connecting point: θ, ϕ, and r. Given three uniform random values $\epsilon_0, \epsilon_1, \epsilon_2$

$$\theta = \epsilon_0 * [0, 2\pi]$$

$$cos(\phi) = \epsilon_1 * [-1, 1]$$

$$r = \epsilon_2^{1/3} * [0, (N/2)\Delta s]$$



$\vec{\boldsymbol{x}}_{N/2}$
$\frac{N}{2}\Delta s$
ϕ
θ
$\vec{\boldsymbol{x}}_0, \vec{\boldsymbol{x}}_N$

Figure 5.12: When the two points $\vec{x}_0$ and $\vec{x}_N$ overlap, point $\vec{x}_{N/2}$ is placed anywhere within the sphere centered at $\vec{x}_0$ with a radius $(N/2)\Delta s$.

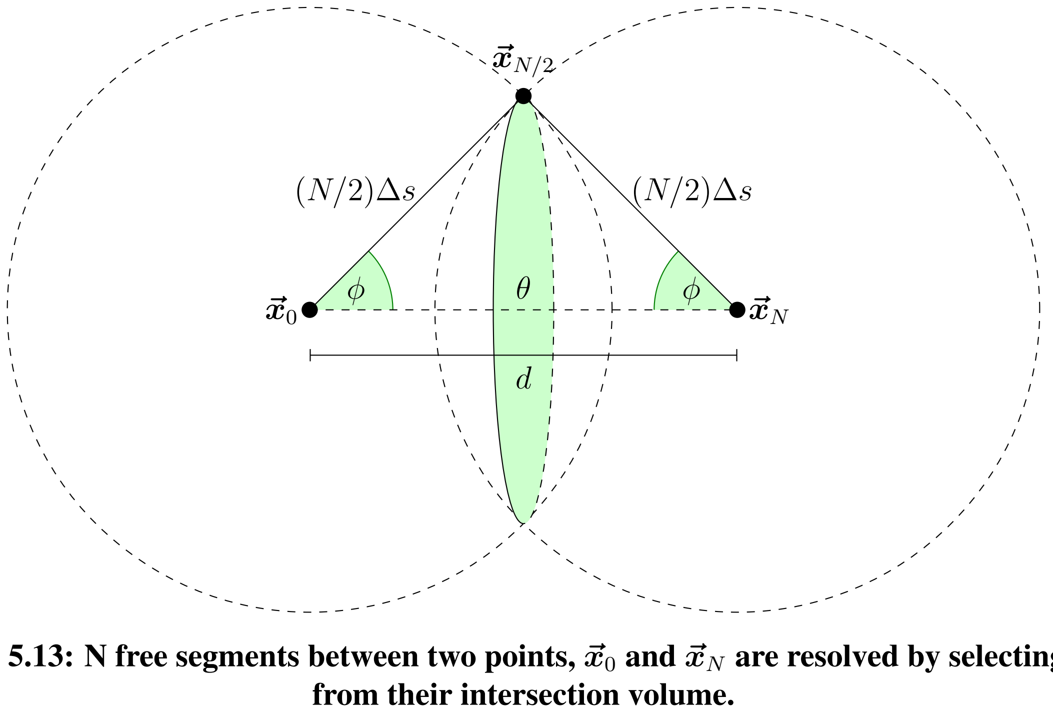


Figure 5.13: N free segments between two points, $\vec{x}_0$ and $\vec{x}_N$ are resolved by selecting $\vec{x}_{N/2}$ from their intersection volume.

In a third case, I have that the two spheres intersect to form a volume region of two stacked spherical caps, as shown in Figure 5.13. For this, I restrict my discussion to that of the spherical cap contained within the sphere centered on $\vec{\boldsymbol{x}}_N$. To fully sample the volume formed by the two caps, I randomly flip the generated $\vec{\boldsymbol{x}}_{N/2}$ point across the intersecting plane connecting the two caps. Like the previous case, I generate three parameters for placing the connecting point: θ, ϕ, and r. Given uniform random values $\epsilon_0, \epsilon_1, \epsilon_2$

$$\theta = \epsilon_0 * [0, 2\pi]$$

$$a = sqrt(\Delta s^2 - (0.5d)^2)$$

$$\phi_{max} = arcsin(a/\Delta s)$$

$$cos(\phi) = \epsilon_1 * [cos(\phi_{max}), 1]$$

$$r = \epsilon_2^{1/3} * \left[\frac{0.5d}{cos(\phi)}, (N/2)\Delta s\right]$$

Using θ, ϕ, and r as spherical coordinates of $\vec{\boldsymbol{x}}_{N/2}$ within the sphere centered on $\vec{\boldsymbol{x}}_0$, I can now place the center position $\vec{\boldsymbol{x}}_{N/2}$, randomly flipping this point across the plane forming the center of the intersecting spherical caps. By sampling and fixing the connection point $\vec{\boldsymbol{x}}_{N/2}$, I can then recurse and repeat the procedure on either half of the free-segment lists, connecting points $\vec{\boldsymbol{x}}_0$ to $\vec{\boldsymbol{x}}_{N/2}$ and points $\vec{\boldsymbol{x}}_{N/2}$ to $\vec{\boldsymbol{x}}_N$. This process recursively continues until I am left with either two free segments, in which they are resolved using the previously discussed two-segment resolution algorithm as a base case, or three free segments, which are resolved using the three-segment resolution algorithm as a second base case.

For this recursive process to be well-formed, with terminating base cases of either two or three segments, this path generation method supports paths only with specific segment counts M in which, for valid, positive integer values n, and $N = M - 2$.

$$(M-2) = 2^n$$

$$or$$

$$(M-2) = 3 * 2^n$$

With this, I have a method of generating unique, random, valid paths of N free segments, which cover the full extent of valid path space from a set of uniformly sampled values between 0 and 1.

5.2.4 Discussion of Path-Generation Sample Bias

For utilization of my path-generation algorithm within a numerical solver, as I will discuss in the next section, it will be important to make clear a very strong assumption which is made regarding the sampling of paths within valid path space.

As discussed, to generate a path with M segments, a number of uniform random samples between 0 and 1 are utilized to uniformly sample θ and ϕ angles along the generated path. As these random values are uniformly sampled from 0 to 1, the process of selecting angle values within their range of values introduces no bias in the final generated path.

Additionally, the independent sampling of each ϕ and radius r is handled in an unbiased manner, as I account for the ϕ and r sampling biases required to sample from a sphere in an unbiased manner. Thus, the individual sampling of each angle on its own does not provide any bias.

However, I find that the angles along the generated path are not sampled independently. For an N free segment path, all angles sampled along the two subdivided free segment lists during the recursive process depend on the selected position of $\vec{x}_{N/2}$. However, for different selections of $\vec{x}_{N/2}$, the volume of possible paths $V(\vec{x}_{N/2})$ changes. Thus, bias is likely introduced, with generated paths not distributed uniformly across valid path space.

In this work, I do not perform further analysis of this bias or the relation between sampling pdfs for different angle selections due to their interdependence during the recurrence process. Instead, I continue with the assumption that with fully uniform random values, I uniformly sample the valid path space. While this assumption was important for completing the remaining work, it is very likely the main cause of bias seen in the remainder of this work.

5.2.5 Path-Generation Numerical Solver

Now that I have a method of generating a unique, valid path $\mathbb{P}$ with M segments that meets the boundary conditions $\mathbb{R}$, I can utilize this to define a novel, path-generation-based solver. Unlike the previously discussed approaches utilizing path-perturbation, the path-generation solver is quite trivial, with the pseudocode shown in Algorithm 3. To solve the path integral $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ using N samples, I simply perform N iterations generating a unique, valid path $\mathbb{P}_i$, weighting the path, and combining the weighted paths into a final sum. Path generation is provided by a uniform random number generator ϵ, from which it can draw uniform samples between 0 and 1.

Algorithm 3 Path Generation Numerical Solver

```
Input: $\mathbb{R} = \{S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\}$
Output: $I$
$I \leftarrow 0$
**for** $i < N$ **do**
    $\epsilon \leftarrow [UniformRandomFloat()]^{2(M-3)+1}\}$
    $\mathbb{P}_i \leftarrow GenerateCurve(M, \mathbb{R}, \epsilon)$
    $I \leftarrow I + W_R(\mathbb{P}_i)$
**end for**
$I \leftarrow I/N$
return $I$
```

The path-generation numerical solver is quite powerful. As discussed previously, the path generation method used allows for uniformly sampling valid path space. In addition, compared to the path perturbation technique, there is no need to generate a seed curve from which all paths are generated through a series of perturbations. The path-generation numerical solver thus avoids any bias introduced in perturbing a path $\mathbb{P}_i$ into $\mathbb{P}_{i+1}$.

5.3 Verification of Numerical Solution With Simplified Model

Ultimately, the best verification of my model would be a comparison to an exact, analytical derivation of the FPI-RT for the number of segments in question. Unfortunately, the radiative transfer weighting kernel is incredibly complex and thus limits the derivation significantly.

However, I am still able to leverage an analytical derivation for validating my numerical solver by utilizing a simplified weighting kernel. As seen in Equation 5.2, the simplified weight path integral formulation, FPI-SW, follows a similar construction to that of FPI-RT, seen in Equation 3.8. However, I utilize a simplified weighting transfer kernel rather than the full radiative transfer weighting kernel. As this simplified kernel depends on the local curvature along the curve and the radiative transfer kernel's scattering σ_b and phase function width μ parameters, I can easily integrate the simplified weighting function into my existing numerical solver.

$$\begin{aligned} G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)_N^{SW} &= \int_\Omega \prod_{i=0}^{N} d\Omega_i \; \delta\left(\hat{\boldsymbol{\beta}}_0 - \hat{\boldsymbol{\omega}}_S\right) \; \delta\left(\hat{\boldsymbol{\beta}}_N - \hat{\boldsymbol{\omega}}_R\right) \\ &\times \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right) \; \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R\right) \\ &\times \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s \sum_{i=0}^{N} \hat{\boldsymbol{\beta}}_i\right) \\ &\times \; \exp\left(-\frac{\alpha}{2} \sum_{i=1}^{N} |\hat{\boldsymbol{\beta}}_i - \hat{\boldsymbol{\beta}}_{i-1}|^2\right) \\ \alpha &= \frac{1}{\sigma_b \mu \Delta s} \end{aligned} \tag{5.1}$$

5.3.1 Deriving the Analytical Solution

The utility of the simplified weighting kernel is revealed when analytically deriving the exact solution of the path integral, starting with paths of minimal segment count. I find that a general, closed form of the transport kernel for up to four free segments within a path can be derived, with a recursive formulation for additional segments to be simple in construction. However, only small segment path counts are reasonable to compute in a feasible amount of time.

By construction, the starting segment with normal $\hat{\boldsymbol{\omega}}_S$ is not included in M. Thus, when targeting a number of path segments M, I set $N = M - 1$. Note that in this formulation, N also represents the number of possible scattering events and weighted segments.

I start by constructing a general form of the Feynman path integral of the simple model. Note that although the form is similar to 3.8, the forms are critically different in their use of different weighting kernels. For this reason, the following analysis using the simplified weighting kernel does not represent full radiative transport. Instead, it is a simplified representation of only absorption and out-scattering along a path and is utilized to explore the numerical solver's behavior independent of the weighting kernel.

As the initial and final directions of propagation, $\hat{\boldsymbol{\beta}}_0 = \hat{\boldsymbol{\omega}}_S$, and $\hat{\boldsymbol{\beta}}_N = \hat{\boldsymbol{\omega}}_R$, are specified as end constraints for the model, their inclusion in the overall integration of the simplified model FPI can be eliminated. The resulting simplified model reduces to the following

$$\begin{aligned} G_N(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \int \prod_{i=1}^{N-1} d\Omega_i \; \Delta s \delta \left(\vec{\boldsymbol{q}} - \sum_{i=1}^{N-2} \hat{\boldsymbol{\beta}}_i - \hat{\boldsymbol{\beta}} \right) \\ \times \prod_{i=1}^{N} \exp\left(-\alpha \left(1 \; - \; \hat{\boldsymbol{\beta}}_i \cdot \hat{\boldsymbol{\beta}}_{i-1} \right) \right) \end{aligned} \tag{5.2}$$

This kernel model lends itself to an iterative construction of G_N from G_{N-1}. Defining the vector

$$\vec{\boldsymbol{q}} = \frac{(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S)}{\Delta s} - (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \tag{5.3}$$

the iterative relation is

$$G_{N+1}(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \exp(-\sigma_a) \int d\Omega'\, G_N(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}', \hat{\boldsymbol{\beta}}')\ \exp(\alpha\, \hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}') \tag{5.4}$$

where the relation models the addition of an additional segment to G_N, accounting for all valid paths which include this new segment by integrating over all possible configurations of the segment. Additionally, the relation accounts for the application of a single absorption-like event $\exp(-\alpha)$ along the newly added segment, and a single scatter-like event $\exp(\alpha\, \hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}')$, whose weight depends on the orientation of the newly added segment.

The lowest term of this construction is

$$G_2(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \frac{\delta(\vec{\boldsymbol{q}})}{(\Delta s)^3} \exp(-\alpha) \exp(\alpha(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{n}}_{\mathbf{0}})) \tag{5.5}$$

This models a discrete curve with two segments and a single scattering event. As there are only two segments, and the start and end directions are constrained, this lowest term naturally takes the form of a delta function.

Using the recursive relation, as well as simplification, G_3 evaluates exactly as

$$G_3(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \frac{\delta(1-q)}{(\Delta s)^3} \exp(-2\alpha) \exp(\alpha \vec{\boldsymbol{q}} \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}})) \tag{5.6}$$

and G_4 is

$$\begin{aligned} G_4(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = {} & \frac{2\pi e^{-4\alpha}}{(\Delta s)^3} \frac{\Theta(2-q)}{q} \\ & \times \exp\left(\frac{\alpha}{2}\ \left(q^2\ +\ \vec{\boldsymbol{q}} \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}})\right)\right) \\ & \times I_0\left(\alpha\sqrt{1-(q^2/4)}\ |\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S|\right) \end{aligned} \tag{5.7}$$

where I_0 is the modified Bessel function of zero order, Θ is the Heaviside step function, and $q = |\vec{\boldsymbol{q}}|$. A full derivation is provided in Appendix D.

Due to the complexity of even the simplest orders of G_N, the direct derivation of higher-order terms of this analytical solution remains out of reach. However, it is trivial to implement a numerical solver of the recursive integral relation to generate a few higher terms of this function, G_5 and G_6. While these obtained results are not an exact analytical solution, they involve an integration over the expanded numerical solution for G_4, and thus act as an important metric for comparison.

$$G_5(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \exp(-\alpha) \int d\Omega' \; G_4(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}', \hat{\boldsymbol{\beta}}') \; \exp(\alpha \, \hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}') \tag{5.8}$$

$$G_6(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) = \exp(-\alpha) \int d\Omega' \; G_5(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}', \hat{\boldsymbol{\beta}}') \; \exp(\alpha \, \hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}') \tag{5.9}$$

Despite being trivial in construction, the numerical evaluation of the recursive relation is expensive computationally, growing exponentially with the number of segments in a path. This currently prevents generating higher-order terms in a reasonable amount of time.

5.3.2 Five Segment Path

I validate my path-generation numerical solver against two different solvers: the analytical derivation-based numerical solver and a numerical integration directly over the free angles. I use the weighting parameters shown in Table 5.1 for all validation experiments.

Table 5.1: Weighting Parameters

σ_a	0.004	Absorption paramter
σ_b	0.1	Scattering paramter
μ	1.0	Phase function width
ϵ	0.075	Regularization paramter

Additionally, I define the following environment, shown in Figure 5.14. I place a beam emitter at the origin, facing down the positive x-axis. I then place a receiver ten units away on the x-axis. However, I allow the receiver to rotate ϕ_R such that $0°$ is no rotation where the emitter and receiver are aligned directionally, $90°$ places the directions perpendicular to each other, and $180°$ faces the receiver directly back toward the emitter.

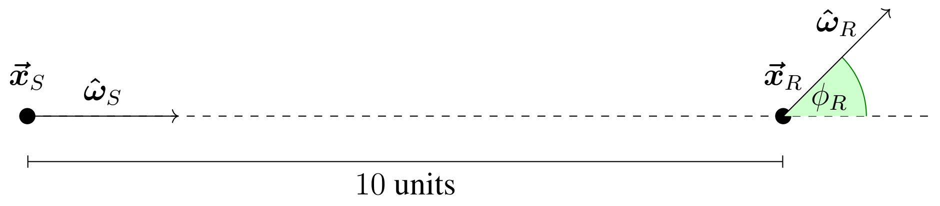


Figure 5.14: The environment construction used for validation of the FPI-SW and FPI-RT numerical solvers. The only parameter which changes in the environment is ϕ_R, which controls the angle of the placed receiver direction $\hat{\boldsymbol{\omega}}_R$. I allow this to vary from facing directly away from the emitter ($0°$) to facing directly back toward the emitter ($180°$).

Lastly, it is important to note that there remains an overall normalization term $\mathbb{D}$ that remains unaccounted for, preventing an exact comparison of numerical data. However, as this normalization acts as a scale factor of the overall FPI solution, I can instead validate my numerical solutions by comparing the relative behavior of the numerical solutions as the receiver angle changes rather than an exact numerical comparison. So long as the two solutions are consistent in their relative behavior, I take this as a valid verification instead of requiring exact, numerical equality.

I first validate my path-generation FPI-SW solver for five-segment paths. I compute a number of FPI-SW solutions for a number of permutations of the validation environment by varying arclength $S = [11, 12, ..., 20]$ and the receiver angle $\phi_R = [0°, 45°, 90°, 135°, 180°]$. By manipulating these dimensions, I explore the numerical solver's behavior in a wide range of cases.

First, I present the results of the analytical derivation numerical solution's execution in this wide array of environments, shown in Table 5.2 and Figure 5.15. Additionally, I evaluate the FPI-SW

Table 5.2: Five Segment Analytical Derivation Results

Arclength	0°	45°	90°	135°	180°
11	4.75E-03	4.62E-03	N/A	N/A	N/A
12	5.04E-04	1.65E-03	N/A	N/A	N/A
13	1.31E-04	5.05E-04	N/A	N/A	N/A
14	6.17E-05	2.32E-04	2.36E-04	N/A	N/A
15	4.23E-05	1.45E-04	1.90E-04	N/A	N/A
16	3.70E-05	1.13E-04	1.54E-04	5.74E-05	N/A
17	3.68E-05	1.01E-04	1.37E-04	9.63E-05	4.00E-05
18	3.72E-05	1.01E-04	1.30E-04	1.05E-04	8.64E-05
19	3.96E-05	1.08E-04	1.32E-04	1.11E-04	1.02E-04
20	4.37E-05	1.23E-04	1.39E-04	1.20E-04	1.14E-04

Table 5.3: Five Segment Angle Integration Solver Results

Arclength	0°	45°	90°	135°	180°
11	4.96E-07	N/A	N/A	N/A	N/A
12	9.52E-09	8.37E-08	N/A	N/A	N/A
13	5.90E-10	5.94E-09	N/A	N/A	N/A
14	7.80E-11	8.17E-10	6.21E-09	N/A	N/A
15	1.74E-11	1.77E-10	1.27E-09	N/A	N/A
16	5.61E-12	5.18E-11	3.52E-10	N/A	N/A
17	2.40E-12	1.98E-11	1.29E-10	4.00E-10	N/A
18	1.34E-12	9.06E-12	5.61E-11	1.51E-10	2.56E-10
19	9.07E-13	4.65E-12	2.69E-11	6.65E-11	1.02E-10
20	6.92E-13	2.71E-12	1.47E-11	3.43E-11	4.97E-11

using a direct, Monte Carlo integration of the FPI-SW over the free parameters θ_0, θ_1, and ϕ_0, which parameterize five segment paths, shown in Table 5.3 and Figure 5.16. Lastly, I evaluate the FPI-SW using my path-generation numerical solver, shown in Table 5.4 and Figure 5.17.

From these results, I draw two main conclusions.

First, I find that the path generation and angle integration numerical solvers converge to the same results. This is an excellent justification for my path-generation method, as it produces the same result as the angle integration, thus demonstrating no bias in my path-generation construction within

Table 5.4: Five Segment Path Generation Solver Results

Arclength	0°	45°	90°	135°	180°
11	4.95E-07	3.47E-06	N/A	N/A	N/A
12	9.51E-09	8.37E-08	N/A	N/A	N/A
13	5.90E-10	5.93E-09	N/A	N/A	N/A
14	7.80E-11	8.16E-10	6.21E-09	N/A	N/A
15	1.74E-11	1.77E-10	1.27E-09	N/A	N/A
16	5.61E-12	5.18E-11	3.52E-10	1.40E-09	N/A
17	2.40E-12	1.98E-11	1.29E-10	4.00E-10	7.90E-10
18	1.34E-12	9.06E-12	5.61E-11	1.51E-10	2.56E-10
19	9.07E-13	4.65E-12	2.69E-11	6.66E-11	1.02E-10
20	6.92E-13	2.71E-12	1.47E-11	3.43E-11	4.97E-11

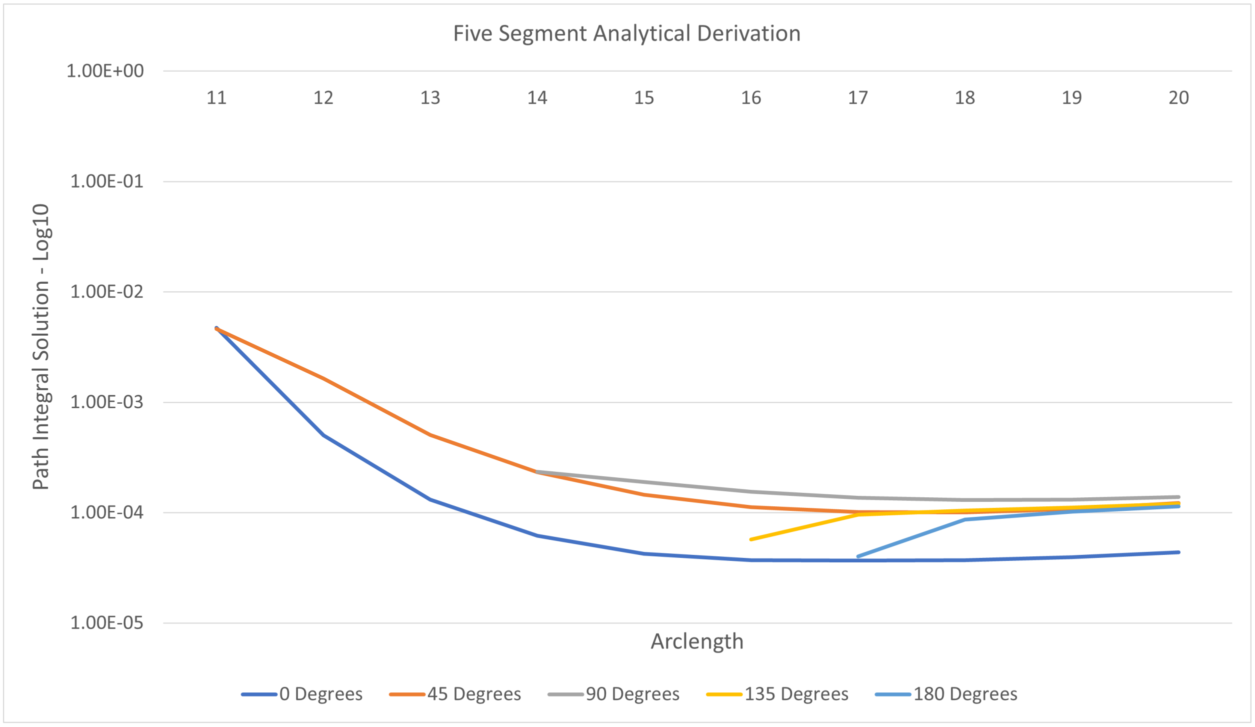


Figure 5.15: Results of the analytical derivation of FPI-SW for a variety of environment configurations and path arclength values for five segment paths.

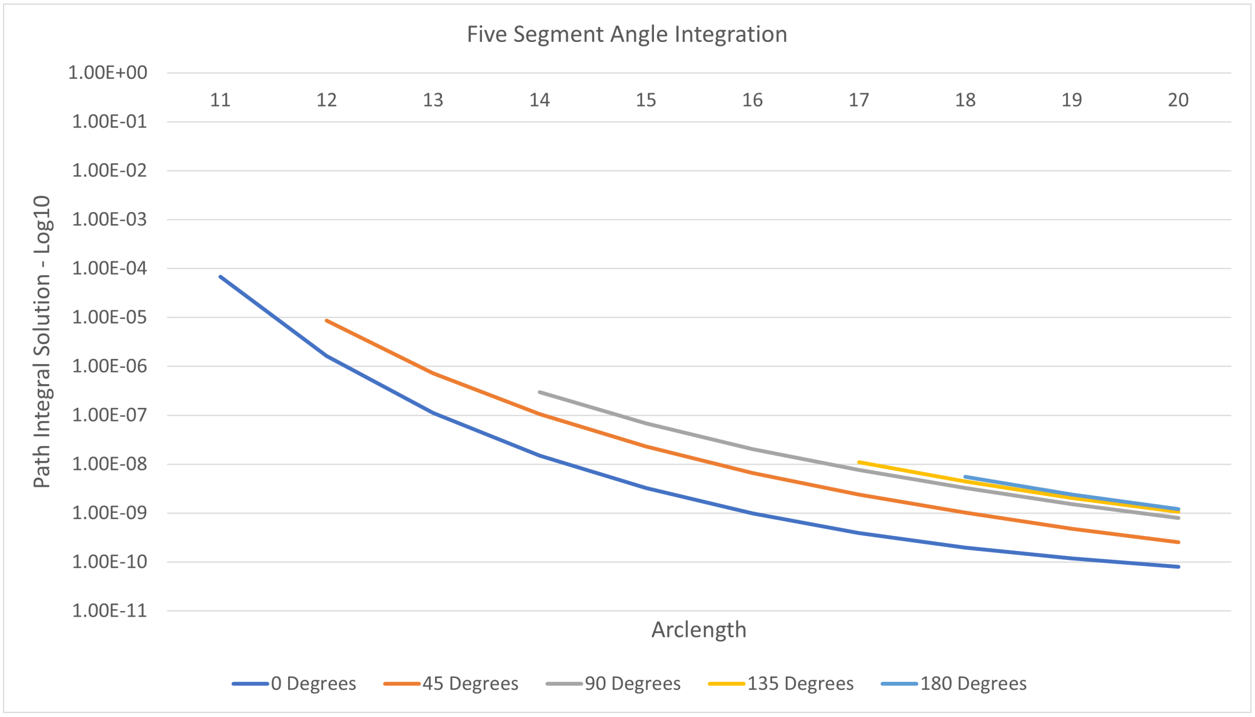


Figure 5.16: Results of angle integration numerical solver of FPI-SW for a variety of environment configurations and path arclength values for five segment paths.

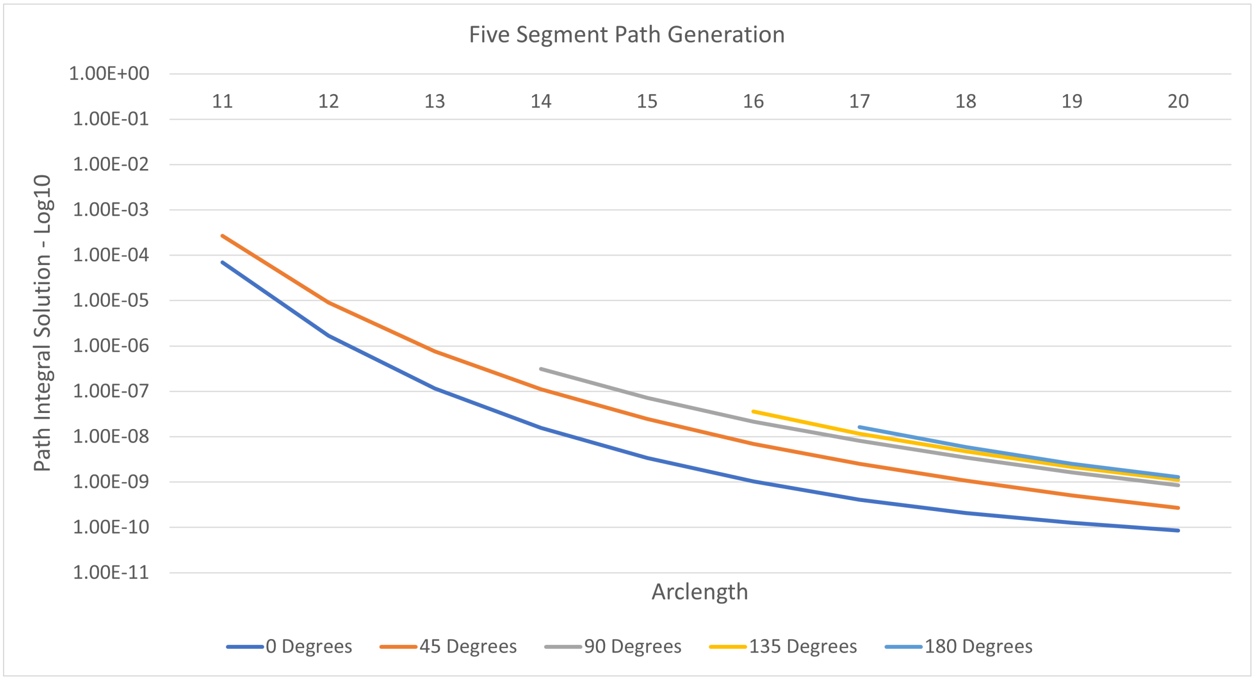


Figure 5.17: Results of path generation numerical solver of FPI-SW for a variety of environment configurations and path arclength values for five segment paths.

the independent angle selections. In addition, my path generation is significantly more efficient, producing 100% valid paths while the angle integration fluctuates around 30% valid paths.

Second, the angle-integration and path-generation solvers differ from the analytical solution. While this difference is limited for the five-segment case, we will find that this trend also continues for the six-segment case, even increasing in severity. I provide a discussion of my best understanding of why this is occurring and my best directions for future work to mitigate these issues after the six-segment data is presented.

5.3.3 Six Segment Path

Next, I present the results of the three numerical solvers in the same environment parameterizations, utilizing a discretization of paths into six segments. First, I demonstrate the analytical derivation numerical results in Table 5.5 and Figure 5.18. It is important to note that with six segments, I am numerically integrating G_5 to obtain G_6, while G_5 is itself numerically integrating the analytically derived G_4. Thus, the analytical derivation is likely less accurate due to this continued abstraction from the direct analytical derivation. Secondly, I present the angle-integration numerical solver from integrating over the angles θ_0, θ_1, θ_2, ϕ_0, and ϕ_1 which parameterize paths of six segments; these results are shown in Table 5.6 and Figure 5.23. Lastly, I show the results of my path-generation numerical solver in Table 5.7 and Figure 5.24. Again, I draw two main conclusions.

First, my path-generation numerical solver again closely matches the direct angle integration. However, in addition to generating 100% valid paths, as opposed to the angle integration's approximately 1% to 30% valid paths, my path-generation numerical solver is able to capture difficult environment configurations that the angle integration struggles to capture. This demonstrates the path-generation numerical solver's ability to integrate over all regions of valid path space, including those for which very narrow parameterizations of paths exist.

Second, my path-generation numerical solver and the angle integration solvers fail to match the

Table 5.5: Six Segment Analytical Derivation Results

Arclength	0°	45°	90°	135°	180°
11	7.70E-04	6.12E-04	N/A	N/A	N/A
12	7.79E-05	2.50E-04	N/A	N/A	N/A
13	1.90E-05	7.82E-05	2.34E-05	N/A	N/A
14	8.52E-06	3.36E-05	3.29E-05	N/A	N/A
15	6.00E-06	1.92E-05	3.22E-05	3.75E-06	N/A
16	5.19E-06	1.40E-05	3.08E-05	7.32E-06	2.91E-06
17	5.51E-06	1.21E-05	3.06E-05	1.03E-05	5.41E-06
18	6.52E-06	1.19E-05	3.16E-05	1.36E-05	7.89E-06
19	8.19E-06	1.28E-05	3.41E-05	1.76E-05	1.10E-05
20	1.06E-05	1.47E-05	3.80E-05	2.28E-05	1.50E-05

Table 5.6: Six Segment Angle Integration Solver Results

Arclength	0°	45°	90°	135°	180°
11	1.59E-10	N/A	N/A	N/A	N/A
12	2.82E-09	7.46E-10	N/A	N/A	N/A
13	6.98E-09	2.81E-09	N/A	N/A	N/A
14	7.43E-09	3.77E-09	1.74E-10	N/A	N/A
15	5.34E-09	3.14E-09	3.05E-10	N/A	N/A
16	3.31E-09	2.08E-09	3.23E-10	2.33E-11	N/A
17	1.90E-09	1.24E-09	2.65E-10	3.29E-11	1.14E-11
18	1.11E-09	7.44E-10	1.98E-10	3.50E-11	1.51E-11
19	6.05E-10	4.18E-10	1.31E-10	3.02E-11	1.50E-11
20	3.28E-10	2.34E-10	8.31E-11	2.35E-11	1.29E-11

Table 5.7: Six Segment Path Generation Solver Results

Arclength	0°	45°	90°	135°	180°
11	6.31E-11	5.46E-12	N/A	N/A	N/A
12	9.44E-10	2.97E-10	N/A	N/A	N/A
13	2.40E-09	1.10E-09	1.60E-11	N/A	N/A
14	2.81E-09	1.54E-09	8.58E-11	N/A	N/A
15	2.28E-09	1.38E-09	1.52E-10	5.07E-12	1.96E-13
16	1.57E-09	9.80E-10	1.68E-10	1.37E-11	3.44E-12
17	9.88E-10	6.26E-10	1.44E-10	1.95E-11	7.14E-12
18	6.16E-10	3.97E-10	1.12E-10	2.11E-11	9.45E-12
19	3.56E-10	2.36E-10	7.70E-11	1.87E-11	9.58E-12
20	2.03E-10	1.39E-10	5.08E-11	1.50E-11	8.45E-12

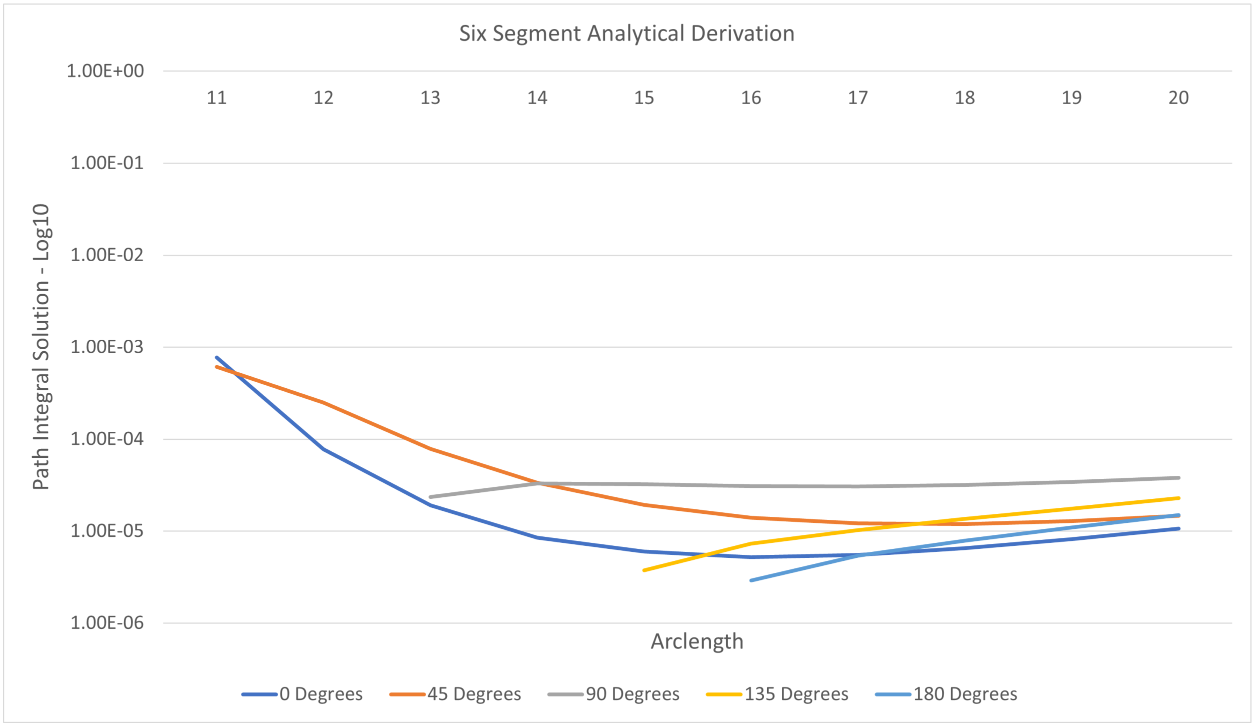


Figure 5.18: Results of the analytical derivation of FPI-SW for a variety of environment configurations and path arclength values for six segment paths.

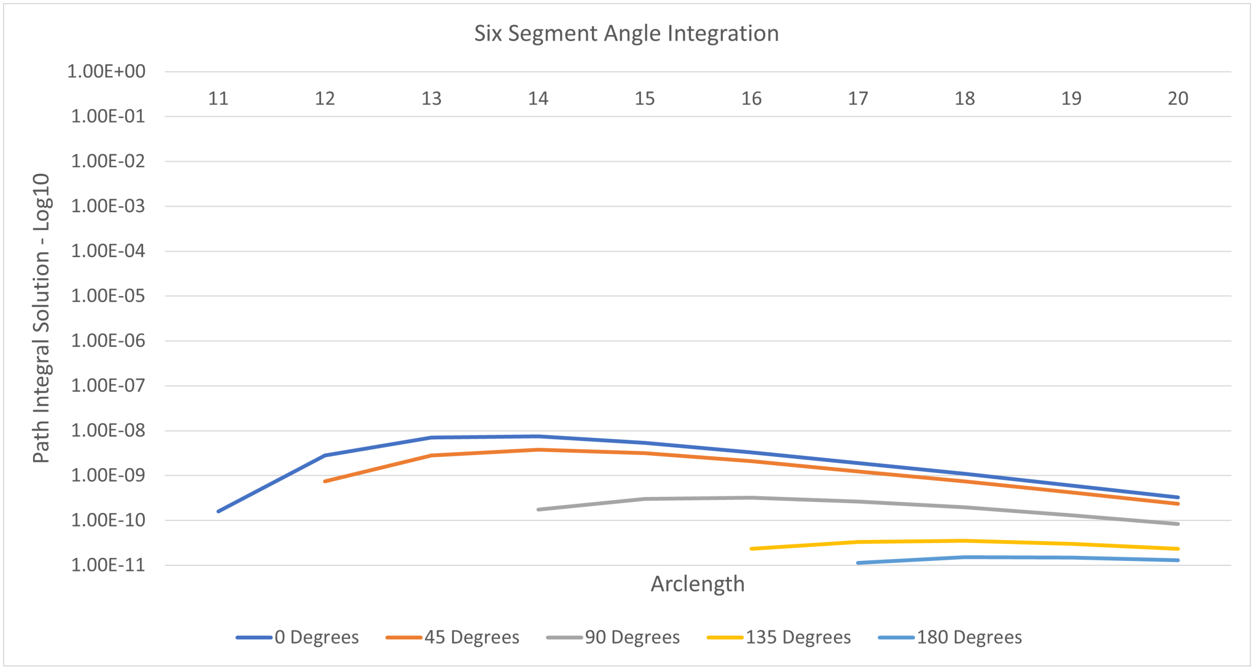


Figure 5.19: Results of angle integration numerical solver of FPI-SW for a variety of environment configurations and path arclength values for six segment paths.

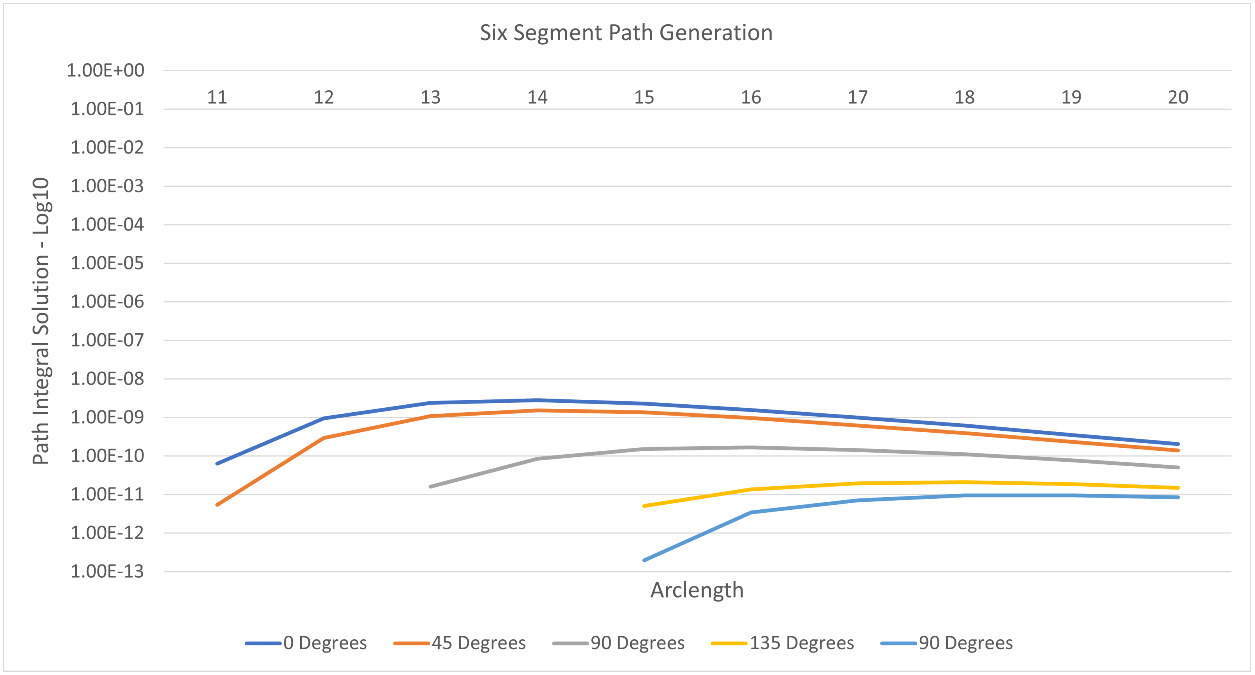


Figure 5.20: Results of path generation numerical solver of FPI-SW for a variety of environment configurations and path arclength values for six segment paths.

analytical derivation. My leading hypothesis for the failure of my path-generation solver to converge to the same solution of the angle integration method is two-fold. First, I believe that a small amount of error is introduced by numerically integrating the analytical solution. However, I believe this integration is minimally impacting the difference in converged solutions.

Our best hypothesis for the divergence in resulting solutions between the analytical derivation solver and that of the angle-integration and path generation solvers is my assumption of uniform valid path space sampling by the path-generation method being incorrect. As I currently assume that valid path space is sampled uniformly, my Monte Carlo estimator within my path-generation numerical solver does not correct for the non-uniform distribution of samples within valid-path space. This same issue plagues the angle integration approach as well. Each angle is treated as independently sampled uniformly from a distribution of possible values in these methods. While this is true for each path as it is being constructed, there is an unaccounted dependence of angles selected later in the process on those selected earlier, particularly the ϕ angles. For example, the selection of ϕ_1, and thus the placement of $\mathbb{S}_2$, depends on the exact selection of ϕ_0 and θ_0, i.e., the placement of $\mathbb{S}_1$. As $\mathbb{S}_1$'s placement changes such that ϕ_1 increases, the volume of space covered by fully exploring θ_1 in $[0, 2\pi]$ differs greatly. Suppose all configurations of $\mathbb{S}_1$ are treated equally from the standpoint of path generation, meaning the same number of path samples are generated for each configuration. In that case, I believe that bias is introduced and valid path space is sampled non-uniformly. Ideally, an analytic representation of the pdf for different angle configurations could be found, allowing direct integration into both the angle-integration and path-generation numerical solvers, potentially correcting this issue.

Despite the discussed shortcomings, I believe that these simplified weighting validation tests demonstrate the power of the path-generation numerical solver. Next, I show the use of my solver for full radiative transfer.

5.4 Verification of Radiative Transfer Numerical Solution

I now verify the use of my path generation numerical solver for use in the full radiative transfer FPI, FPI-RT. Unfortunately, when calculating the FPI-RT, and thus using the full radiative transfer weighting kernel, a direct comparison to ground truth is impossible as I cannot analytically derive the small segment solution of the FPI-RT with this more complex weighting kernel. Instead, I validate the path-generation numerical solver in two ways. First, I show that the angle-based numerical solver and path-generation numerical solvers provide the same results, as seen previously in the simplified weight section. Then, I additionally show that for increasing path counts, the path-generation numerical solver converges into a stable solution for a variety of environments as path count increases.

5.4.1 Angle Integration and Path Generation Comparison

I begin with a comparison of the angle integration numerical solver and path generation numerical solvers for the full FPI-RT for the same environment construction and parameterization as the simple weight verification, shown in Figure 5.14.

Table 5.8: FPI-RT Five-Segment Angle Integration Solver Results

Arclength	0°	45°	90°	135°	180°
11	3.44E-03	N/A	N/A	N/A	N/A
12	1.36E-03	6.01E-02	N/A	N/A	N/A
13	8.48E-04	6.54E-02	N/A	N/A	N/A
14	6.58E-04	7.48E-03	6.04E-02	N/A	N/A
15	6.26E-04	5.03E-03	3.73E-02	N/A	N/A
16	9.26E-04	6.36E-03	2.19E-02	N/A	N/A
17	8.57E-04	5.66E-03	1.55E-02	7.78E-02	N/A
18	3.81E-04	1.43E-03	1.20E-02	2.82E-02	6.49E-03
19	2.45E-04	2.38E-04	9.06E-03	1.20E-02	7.91E-04
20	1.94E-04	1.92E-04	6.57E-03	6.76E-03	5.04E-04

Table 5.9: FPI-RT Five-Segment Path Generation Solver Results

Arclength	0°	45°	90°	135°	180°
11	3.45E-03	2.01E+00	N/A	N/A	N/A
12	1.36E-03	6.01E-02	N/A	N/A	N/A
13	8.48E-04	6.59E-02	N/A	N/A	N/A
14	6.59E-04	7.49E-03	6.03E-02	N/A	N/A
15	6.27E-04	5.06E-03	3.77E-02	N/A	N/A
16	9.26E-04	6.33E-03	2.23E-02	4.02E-01	N/A
17	8.56E-04	5.62E-03	1.57E-02	7.83E-02	7.46E-01
18	3.81E-04	1.41E-03	1.22E-02	2.82E-02	6.51E-03
19	2.45E-04	2.38E-04	9.21E-03	1.21E-02	7.92E-04
20	1.93E-04	1.92E-04	6.62E-03	6.83E-03	5.05E-04

Focusing first on five-segment paths, the angle integration numerical solver results are shown in Table 5.8 and Figure 5.21. Likewise, the path generation numerical solver results are shown in Table 5.9 and Figure 5.22. For five segment paths, the results of both numerical solvers closely match, with the path generation solver exploring more valid paths and thus providing a result for more environment configurations than the angle integration solver, which fails to discover valid paths for these specific configurations. Similarly to the simplified weight validation experiments, my path generation numerical solver has all valid samples during path generation and thus performs significantly better than the angle integration solver.

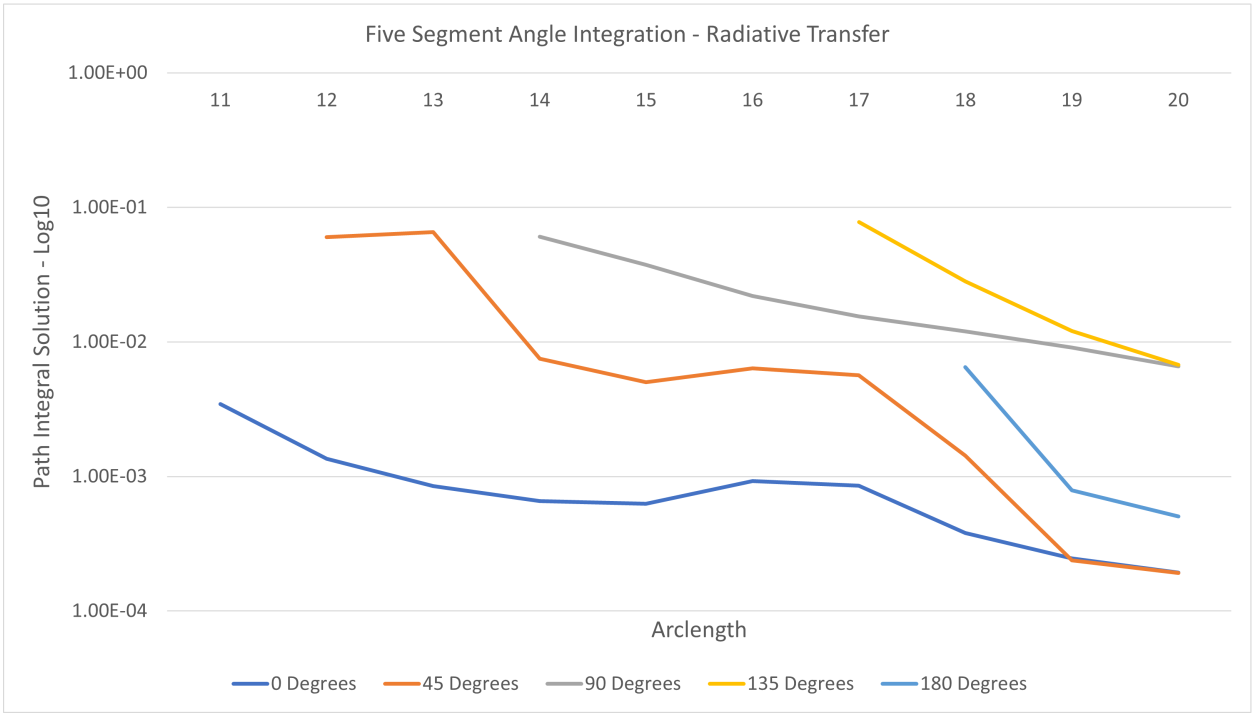


Figure 5.21: Results of angle integration numerical solver of FPI-RT for a variety of environment configurations and path arclength values for five segment paths.

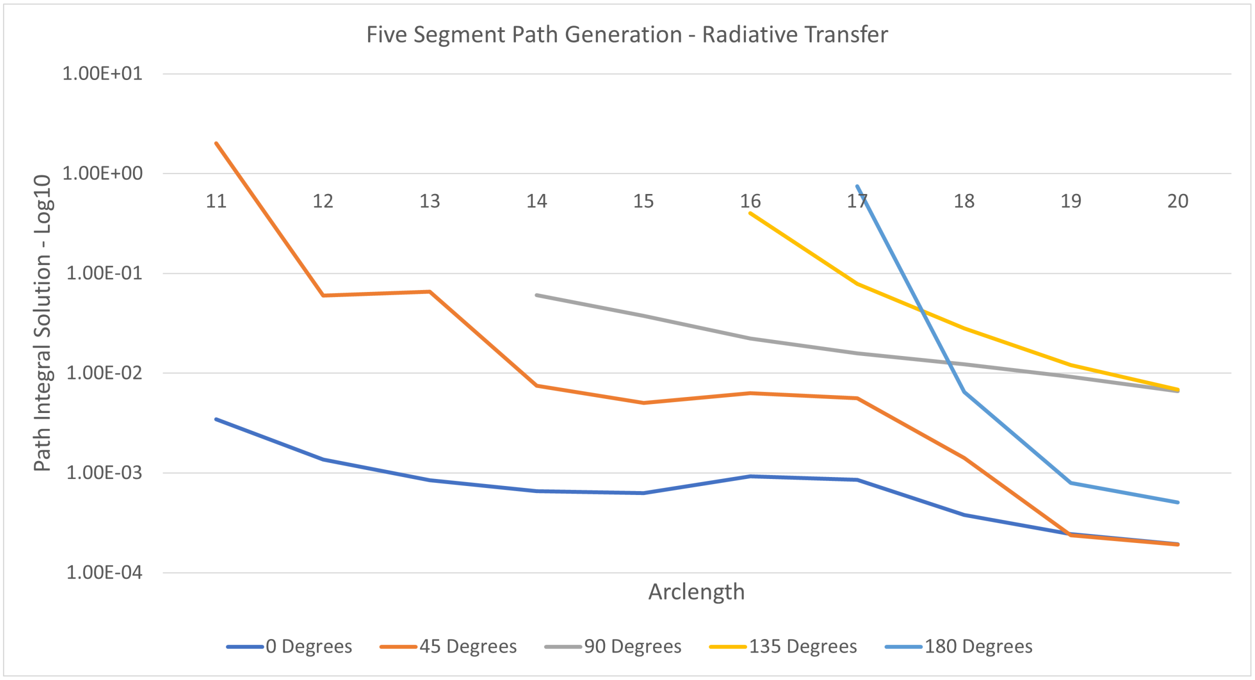


Figure 5.22: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values for five segment paths.

Table 5.10: FPI-RT Six-Segment Angle Integration Solver Results

Arclength	0°	45°	90°	135°	180°
11	N/A	N/A	N/A	N/A	N/A
12	1.03E-03	N/A	N/A	N/A	N/A
13	6.16E-04	1.79E-04	N/A	N/A	N/A
14	6.19E-04	2.19E-04	N/A	N/A	N/A
15	8.15E-04	1.97E-04	0.00E+00	N/A	N/A
16	3.39E-04	1.68E-04	7.86E-05	0.00E+00	N/A
17	1.85E-04	1.25E-04	6.90E-05	0.00E+00	0.00E+00
18	1.45E-04	9.85E-05	6.58E-05	0.00E+00	0.00E+00
19	1.27E-04	8.26E-05	5.78E-05	6.57E-05	0.00E+00
20	1.21E-04	7.38E-05	5.20E-05	5.98E-05	6.69E-05

Table 5.11: FPI-RT Six-Segment Path Generation Solver Results

Arclength	0°	45°	90°	135°	180°
11	4.56E-02	5.92E+00	N/A	N/A	N/A
12	1.41E-02	1.19E-01	N/A	N/A	N/A
13	1.25E-02	2.56E-02	2.29E+00	N/A	N/A
14	1.25E-02	9.74E-02	5.72E-02	N/A	N/A
15	3.58E-02	3.94E-01	2.00E-02	7.23E-01	N/A
16	9.87E-03	7.81E-01	1.37E-02	4.49E-02	9.44E-02
17	9.23E-03	1.02E+00	1.14E-02	1.88E-02	1.00E-02
18	8.97E-03	1.12E+00	1.34E-02	1.36E-02	4.84E-03
19	8.36E-03	9.87E-01	1.22E-02	1.16E-02	3.93E-03
20	7.74E-03	7.75E-01	6.26E-03	1.14E-02	3.83E-03

Next, focusing on six-segment paths, the angle integration numerical solver results are shown in Table 5.10 and Figure 5.23. Likewise, Table 5.11 and Figure 5.24 show the path generation numerical solver results. For six segment paths, the results of the numerical solvers significantly diverge. With this more complex weighting function, the angle integration method struggles to discover paths that contribute significantly to the FPI solution; approximately 1% of path samples are valid. In contrast, the path generation solver explores only valid paths and thus performs significantly better. I believe that the difference in the two solver results shows the strength of the path generation numerical solver for generating a solution to FPI-RT.

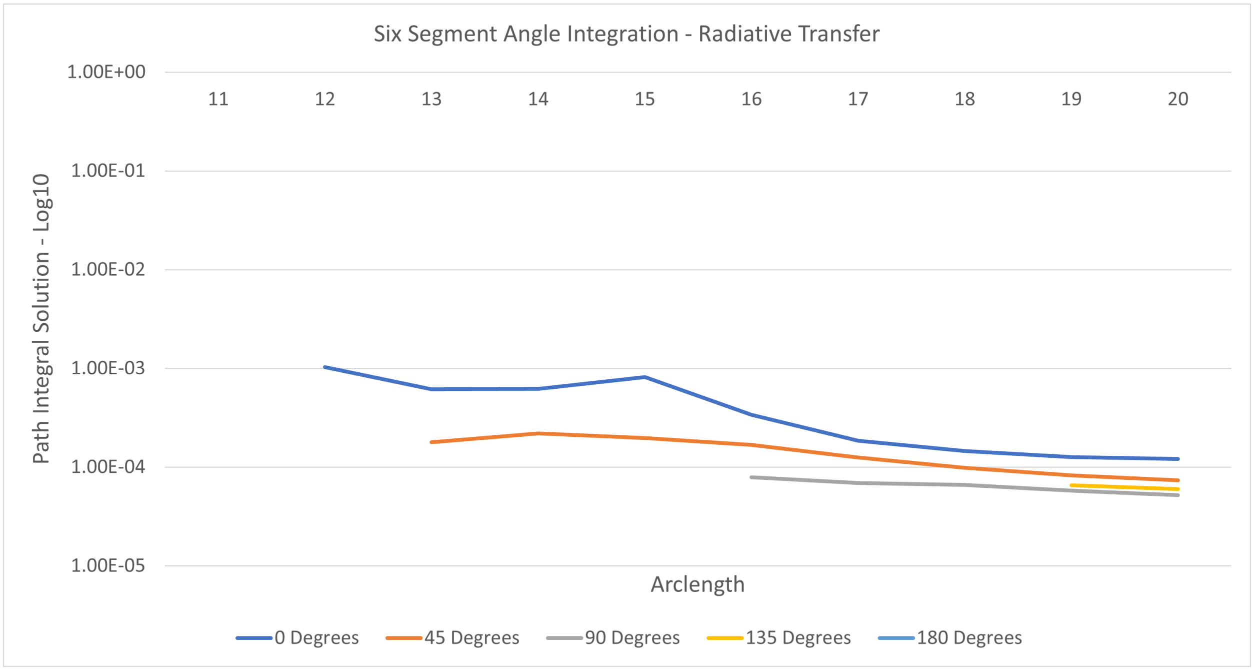


Figure 5.23: Results of angle integration numerical solver of FPI-RT for a variety of environment configurations and path arclength values for six segment paths.

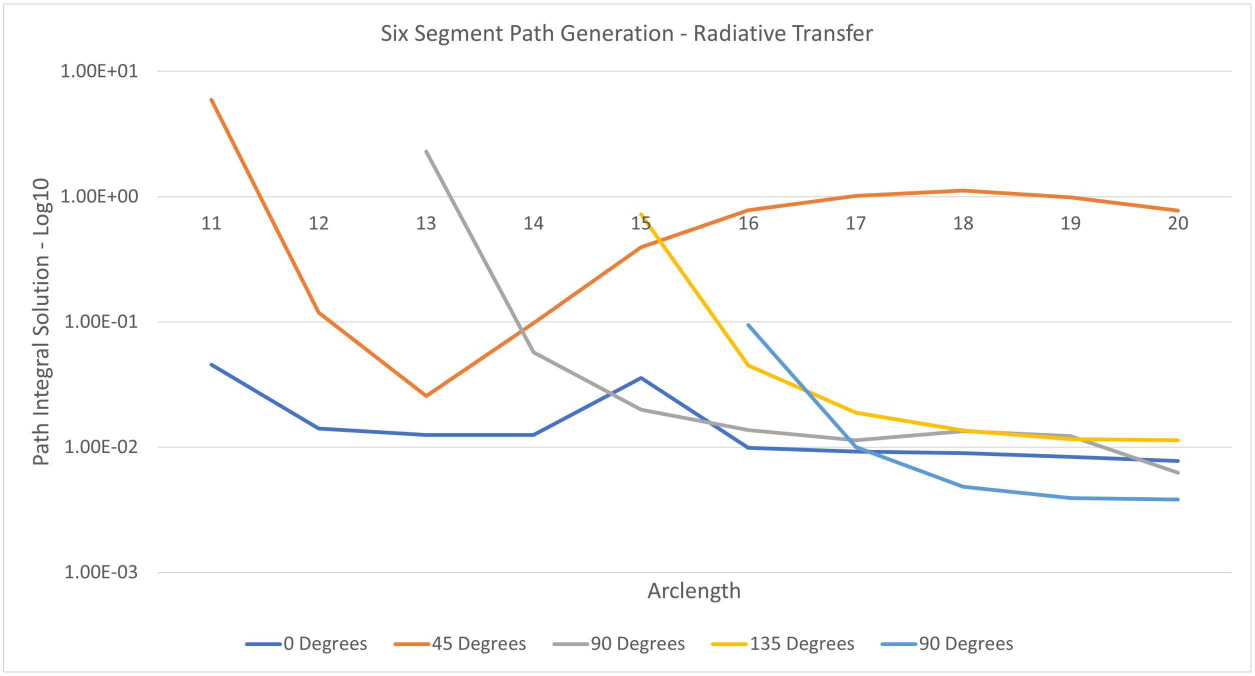


Figure 5.24: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values for six segment paths.

5.4.2 Path Generation Convergence for High Segment Counts

Additionally, I want to show that the path generation numerical solver for FPI-RT converges for higher segment counts. For this, I selected paths of 66 segments (with 64 free segments) for a few reasons.

1. 66 segments allows for a large orders of scatter, one per segment. As the power of the FPI-RT is the ability to capture high orders of scattering, I want to select as high a segment count as possible.

2. In addition to allowing for more scattering, higher segment counts geometrically allow for more path configurations. Thus, I want to select as high a value as possible, weighed against my third criteria.

3. Due to the lack of FPI normalizer, the overall FPI-RT solution grows with the number of segments. 66-segments was the largest path segment count I found which allowed for data visualiation of the calculated FPI-RT. While I can efficiently simulate higher segment count FPI-RT solves, visualizing the data quickly becomes painful as the final FPI solution outgrows common floating point representations.

Utilizing the same environment parameterization used in the previous verification experiments, I explore the convergence of the path generation numerical solver as the number of paths per FPI-RT invocation increases from 10^6 to 10^9.

As seen in the results in Tables 5.12 to 5.15 and Figures 5.25 to 5.28, the path generation solution converges quickly to a converged solution, even with as few as 10^6 paths. As the path count increases, one can see moderate noise and fluctuation, such as the bump shown for the $135°$ receiver path with 10^8 paths. However, with the increase in paths, the solution converges toward the expected, stable solution, and these fluctuations stabilize.

Additionally, Table 5.16 compares the performance of the path generation solver to that of the

Table 5.12: FPI-RT 66-Segment, 10^6 Paths

Arclength	0°	45°	90°	135°	180°
11	2.06E+124	1.51E+127	1.77E+124	2.04E+124	2.76E+124
12	5.86E+123	6.99E+125	3.14E+123	1.90E+123	1.57E+123
13	3.94E+123	7.88E+126	2.19E+123	1.31E+123	1.12E+123
14	3.06E+123	1.19E+127	1.85E+123	1.14E+123	9.05E+122
15	2.47E+123	8.13E+126	1.56E+123	9.72E+122	7.98E+122
16	2.24E+123	4.05E+126	1.54E+123	9.54E+122	7.60E+122
17	2.08E+123	1.60E+126	1.50E+123	9.45E+122	7.47E+122
18	2.05E+123	4.11E+125	1.48E+123	9.64E+122	7.94E+122
19	1.97E+123	1.25E+125	1.47E+123	9.74E+122	8.06E+122
20	2.01E+123	3.74E+124	1.51E+123	1.03E+123	8.42E+122

Table 5.13: FPI-RT 66-Segment, 10^7 Paths

Arclength	0°	45°	90°	135°	180°
11	2.18E+124	1.38E+127	1.42E+124	2.14E+124	2.77E+124
12	5.81E+123	7.20E+125	3.12E+123	1.94E+123	1.50E+123
13	3.84E+123	7.29E+126	2.19E+123	1.33E+123	1.05E+123
14	2.98E+123	1.27E+127	2.03E+123	1.12E+123	8.91E+122
15	2.53E+123	8.50E+126	1.62E+123	1.00E+123	8.01E+122
16	2.27E+123	3.66E+126	1.52E+123	9.59E+122	7.76E+122
17	2.11E+123	1.27E+126	1.47E+123	9.65E+122	7.69E+122
18	2.04E+123	3.93E+125	1.48E+123	1.02E+123	7.88E+122
19	1.99E+123	1.19E+125	1.50E+123	1.06E+123	8.20E+122
20	1.96E+123	3.50E+124	1.50E+123	1.02E+123	8.29E+122

Table 5.14: FPI-RT 66-Segment, 10^8 Paths

Arclength	0°	45°	90°	135°	180°
11	2.05E+124	1.29E+127	1.52E+124	2.14E+124	2.77E+124
12	5.80E+123	7.10E+125	3.14E+123	1.90E+123	1.51E+123
13	3.84E+123	7.34E+126	2.20E+123	1.33E+123	1.05E+123
14	3.00E+123	1.26E+127	2.17E+123	1.12E+123	8.92E+122
15	2.51E+123	8.50E+126	3.43E+123	1.00E+123	7.95E+122
16	2.25E+123	3.68E+126	1.52E+123	9.66E+122	7.69E+122
17	2.12E+123	1.27E+126	1.48E+123	9.61E+122	7.74E+122
18	2.04E+123	4.01E+125	1.48E+123	9.88E+122	7.91E+122
19	2.04E+123	1.23E+125	1.51E+123	6.51E+123	8.58E+122
20	1.95E+123	3.67E+124	1.53E+123	1.03E+123	8.34E+122

Table 5.15: FPI-RT 66-Segment, 10^9 Paths

Arclength	0°	45°	90°	135°	180°
11	2.10E+124	1.20E+127	1.50E+124	2.16E+124	2.80E+124
12	5.80E+123	7.09E+125	3.13E+123	1.89E+123	1.52E+123
13	3.84E+123	7.36E+126	2.20E+123	1.33E+123	1.06E+123
14	3.00E+123	1.26E+127	2.65E+123	1.12E+123	8.93E+122
15	2.51E+123	8.47E+126	2.09E+123	9.98E+122	7.95E+122
16	2.26E+123	3.69E+126	1.54E+123	9.63E+122	7.71E+122
17	2.12E+123	1.28E+126	1.50E+123	9.61E+122	7.73E+122
18	2.04E+123	4.01E+125	1.48E+123	1.12E+123	7.92E+122
19	2.01E+123	1.23E+125	1.51E+123	2.36E+123	8.30E+122
20	1.95E+123	3.61E+124	1.51E+123	1.05E+123	8.66E+122

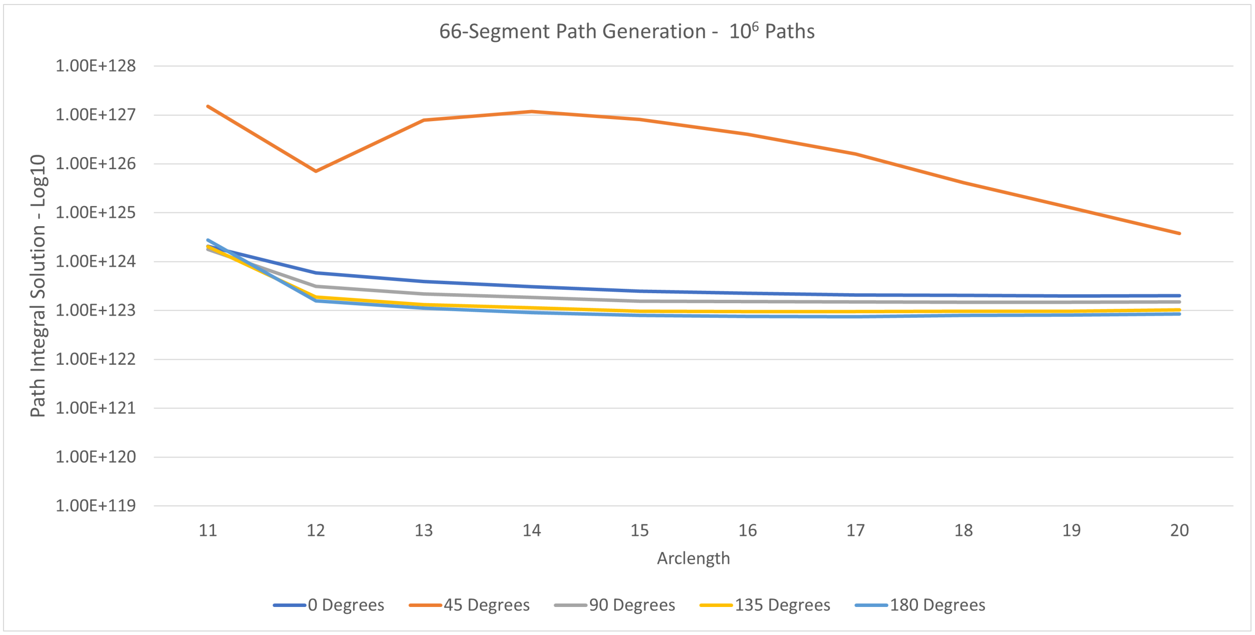


Figure 5.25: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values. This experiment simulates paths with 66 segments, generating 10^6 path samples.

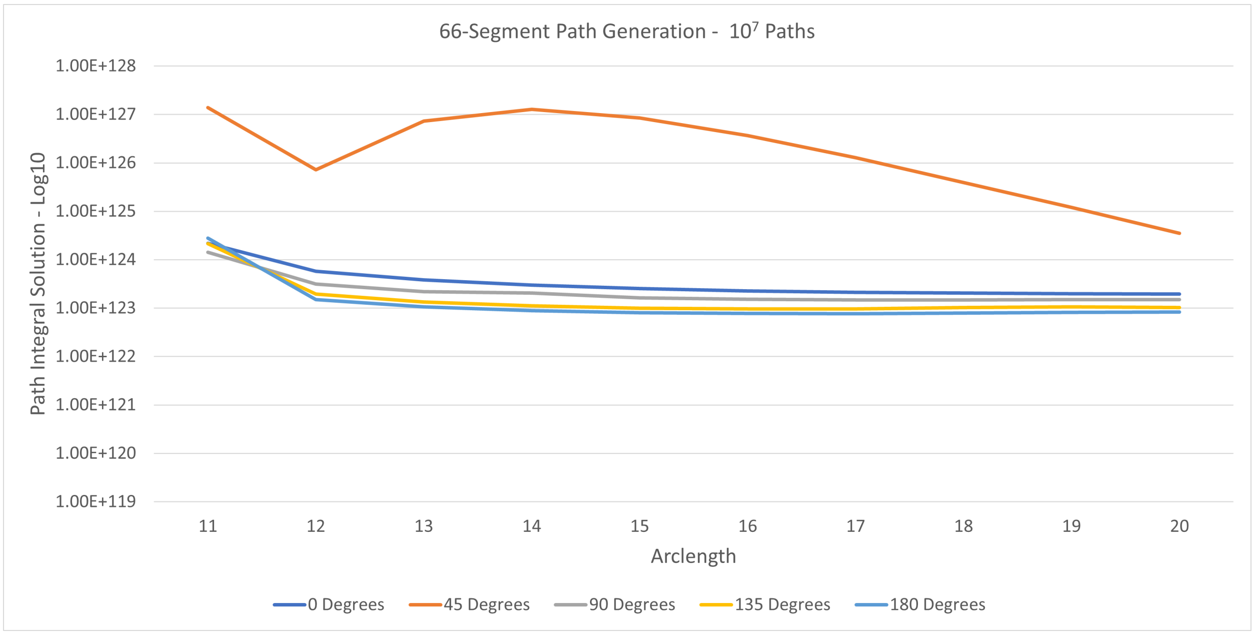


Figure 5.26: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values. This experiment simulates uses paths with 66 segments, generating 10^7 path samples.

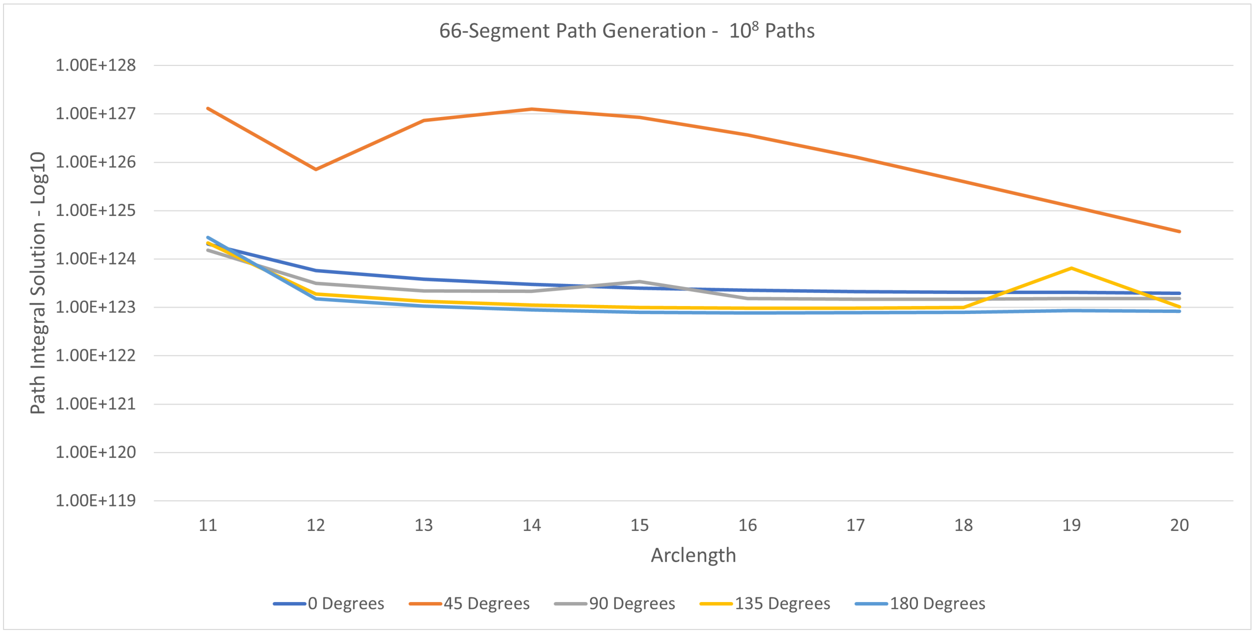


Figure 5.27: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values. This experiment simulates uses paths with 66 segments, generating 10^8 path samples.

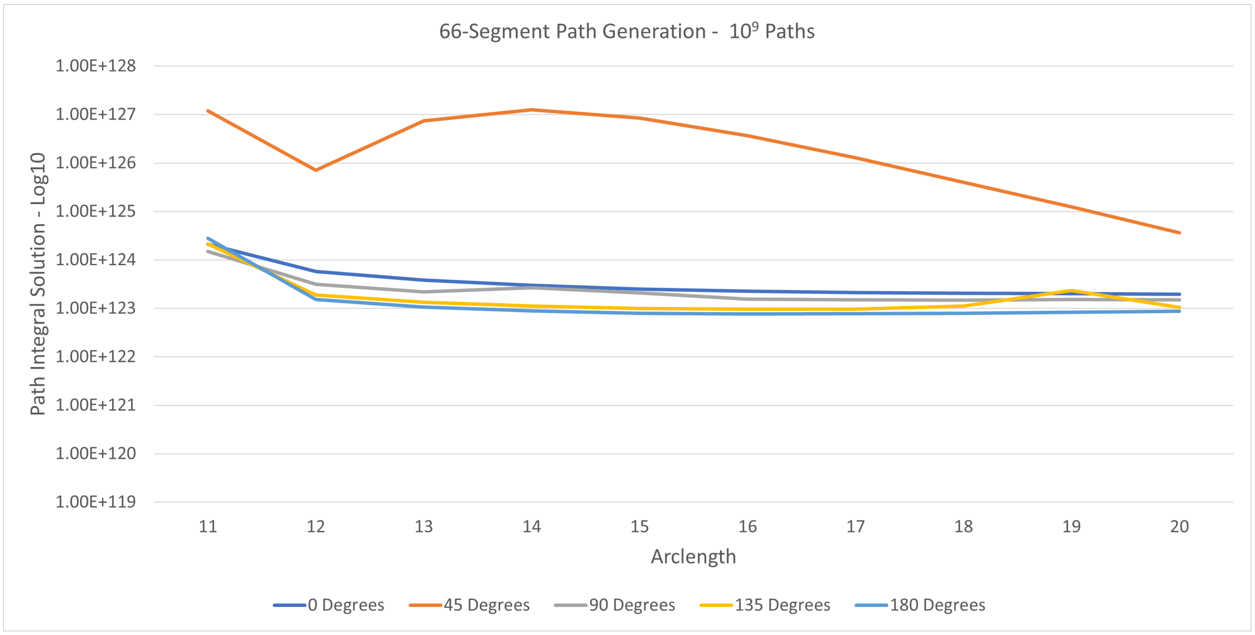


Figure 5.28: Results of path generation numerical solver of FPI-RT for a variety of environment configurations and path arclength values. This experiment simulates uses paths with 66 segments, generating 10^9 path samples.

Table 5.16: Path Generation Performance

Path Count	Path Generation (min)	Path Perturb (min)
10^6	0.08	0.23
10^7	1.02	1.31
10^8	7.67	12.20
10^9	72.8	121.88

path perturbation solver on the CPU. These are measured the same as Table 4.1, using the path generation solver instead. Experimentation finds the path generation technique performs 1.3x to 1.5x better than the path perturbation solver. It is also important to note that the path generation technique remains four orders of magnitude times faster than previous FPI-RT solvers.

Again, it is essential to note that these FPI-RT results are obtained assuming that valid path space is uniformly explored. However, as previously discussed, this is unfortunately not likely the case. Thus I expect that as a correction for uniform sampling of valid path space is added to the path generation numerical solver, the converged results could vary significantly. Still, the relative behavior and, thus, conclusions drawn from the angle integration and path generation numerical solver comparisons will be maintained as the uniform sampling of valid path space assumption is made for both solvers and will impact both methods in the same manner.

6. APPLICATION OF MODEL TO VIRTUAL ENVIRONMENT

In the context of computer graphics, I desire to render an image of a virtual environment containing a volume using my FPI-RT numerical solver. In the following sections, I demonstrate a number of applications of my proposed path-generation numerical solver to generate images of homogeneous and non-homogeneous environments.

6.1 Generalization of Path Integral to Full Environment

To utilize the presented numerical solvers for rendering an image of a full environment, it is essential to start by discussing exactly what the numerical solver provides.

I start with $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, representing the path integral. The keen reader will note that while useful, this formulation is quite limited in its applicability to capture complete radiative transfer in a virtual environment, as the FPI-RT is missing key integrations over all possible arclengths of light travel paths, emitter positions, and directions. However, I can utilize the FPI-RT within a greater integration over other environmental parameters to obtain a complete solution.

The obvious next step for the application would involve additional integration over all possible lengths of paths, as well as emitter locations and directions, shown in Equation 6.1. This can be efficiently integrated into the path-generation numerical solver by modifying the path generator to be parameterized by arclength. Thus, when integrating over all possible paths and randomly generating paths, a random arclength between S_{min} and S_{max} is drawn and used to specify the boundary conditions of the corresponding path sample.

$$L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = \int_{S_{min}}^{S_{max}} G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)\, dS \tag{6.1}$$

To numerically compute this integration, I need to compute reasonable values for S_{min} and S_{max}.

6.1.1 Minimum Arclength Limit

First, I discuss the minimum limit for the arclength integration. Moving forward, I assume no opaque geometry; i.e., paths are never prevented from propagating forward by reflecting off opaque surfaces but rather scattered due to a local phase function. In order to successfully compute the path integral for a given set of boundary conditions, I need an exact way of finding the minimum arclength path that meets the boundary condition. I show a derivation of this formula in Equation 6.2.

$$
\begin{aligned}
(M-2)\Delta s &= \left|(\vec{\boldsymbol{x}}_R - \Delta s\hat{\boldsymbol{\omega}}_R) - (\vec{\boldsymbol{x}}_S + \Delta s\hat{\boldsymbol{\omega}}_S)\right| \\
(M-2)\frac{S}{M} &= \left|(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) - \frac{S}{M}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right)\right| \\
\left((M-2)\,\frac{S}{M}\right)^2 &= \left|(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) - \frac{S}{M}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right)\right|^2 \\
\frac{S^2}{M^2}(M^2-4M+4) &= \left|(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) - \frac{S}{M}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right)\right|^2 \\
\frac{S^2}{M^2}(M^2-4M+4) &= \left((\vec{\boldsymbol{x}}_R - \hat{\boldsymbol{\omega}}_S) - \frac{S}{M}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right)\right) \\
&\quad \cdot \left((\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) - \frac{S}{M}(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R)\right) \\
\frac{S^2}{M^2}(M^2-4M+4) &= (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad - \frac{2S}{M}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad + \frac{S^2}{M^2}\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \\
S^2(M^2-4M+4) &= M^2\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S\right) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad - 2SM\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad + S^2\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \\
0 &= M^2\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S\right) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad - M2S\left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
&\quad + S^2\left((\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) - \left(M^2-4M+4\right)\right)
\end{aligned}
\tag{6.2}
$$

Using this formulation, I can solve for S_{min} using the quadratic formula with the following parameters.

$$
\begin{aligned}
a &= (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \cdot (\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) - \left(M^2 - 4M + 4\right) \\
b &= -2M\,(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \\
c &= M^2\,(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S) \cdot (\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S)
\end{aligned}
$$

I calculate each candidate of minimum arclength, S_0, S_1, and select the minimum, a positive candidate from the two possible candidates to be S_{min}.

Again, this construction is valid only for environments without opaque geometry. If there were to be any boundaries formed by opaque geometry within the environment, this technique does not consider them. It could produce a minimum arclength path that would intersect with this boundary and thus be invalid.

6.1.2 Maximum Arclength Limit

Next, I discuss the upper limit of S_{max}. It is trivial to see that there is no maximum arclength for a path light could travel. Thus, a full theoretical evaluation of the FPI-RT in an environment would require integration to $S_{max} = \infty$. However, in practice, longer paths will contribute less and less to the overall amount of light received at the end position due to two factors. First, any environmental absorption acts to downweigh the path exponentially with the path length, $exp(\sigma_a S)$, leading to longer paths providing exponentially less contribution than shorter ones. Secondly, in most realistic scenarios, paths are weighted such that those with greater curvature tend to be weighed lower than those with less curvature. When maintaining the same start and end positions and directions but allowing arclength to change, it is trivial to see that the curvature of a path increases as the path length increases. Thus longer paths are weighted lower than shorter paths.

Note that this is a general trend and not an absolute truth, as it is possible to construct a path of greater length that is weighted higher than a shorter path. However, these are generally unlikely cases; thus, the general trend holds. In practice, I limit my numerical solver to a reasonable upper limit, S_{max}, which I experimentally determined to work well when set to four times the S_{min}.

6.1.3 Discussion of Arclength Sampling Assumption

Similar to my uniform sampling of valid path space assumptions discussed during path generation, a similar assumption is made for arclength sampling. For the results presented in this work, I uniformly sample arclengths between S_{min} and S_{max} and assume all samples to contribute equally to the final path solution.

However, I hypothesize that this assumption is not correct. As arclength increases, the volume of valid path space increases as well. With this, an equal number of samples for S_{min} and S_{max} would lead to a significant bias toward lower arclength paths in the full experiment's valid path space. To correct for this, samples should be biased toward paths with larger arclengths.

However, while more samples should be taken for larger arclengths, the radiative transfer weighting significantly downweighs longer paths. Thus samples with larger arclength would contribute significantly less to the overall solution. Thus, to aid in convergence, a practical numerical solution would want to utilize an importance sampling scheme to sample lower-weighted paths. Uniform sampling of the arclength range does something similar to this while missing the key sample weight correction required for an importance sampling scheme. To correct this, the pdf of arclength with regards to full experiment path space would need to be discovered and integrated into the numerical solver. This is later discussed in the future work section of the conclusion.

6.2 Rendering Beam Spread on Light Field Receiver With FPI-RT

I start by first showing the application of my method to a trivial environment and building from there. I select a simple beam spread environment, shown in Figure 6.1. In this environment, a beam emitter is placed at the origin, oriented to emit down the positive z-axis. I then place a receiving image located ten units along the z-axis, oriented along the XY-plane, facing the emitter. I subdivide this image plane into several evenly sized pixels and model the amount of light received

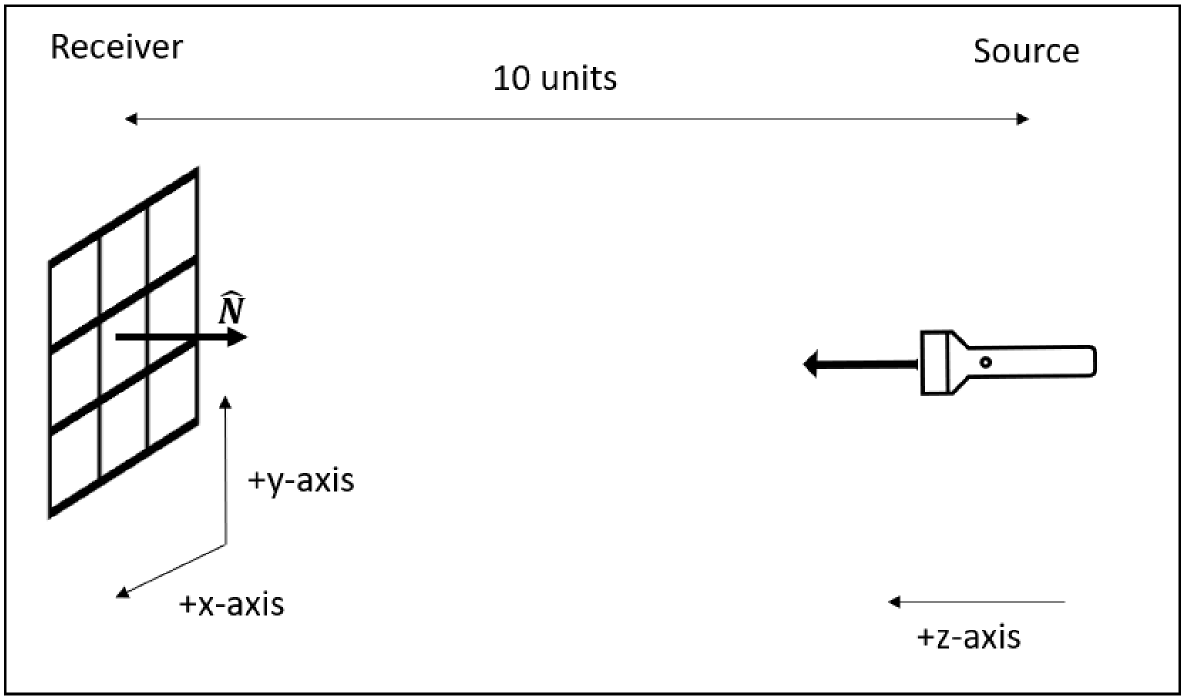


Figure 6.1: Empty environment configuration. Note, the full environment is considered to consist of homogenous, low-scattering, and low-absorption material.

by each pixel as the weighted average of light received at the center of the pixel from N directions along the hemisphere centered around the plane's normal.

For this experiment, I model the environment as consisting of a homogeneous, low-scattering medium, with a light emitter and receiving plane placed symmetrically around the z-axis. With this construction, I expect the received light detected by the image place to be symmetrical around the z-axis (more specifically, the distance of the pixel center to the z-axis). Thus, symmetry in my results verifies the convergence of my path generation numerical solver.

I demonstrate this falloff in several beam-spread environments with a volume of absorption $\sigma_a = 0.04$, a scattering parameter of $\sigma_b = 0.1$, and paths containing 66 segments per the same reasons as my FPI-RT validation. Additionally, the images shown here are a log plot of the FPI-RT result, as this more accurately captures human perception, which are then normalized to the $[0, 1]$ range for display.

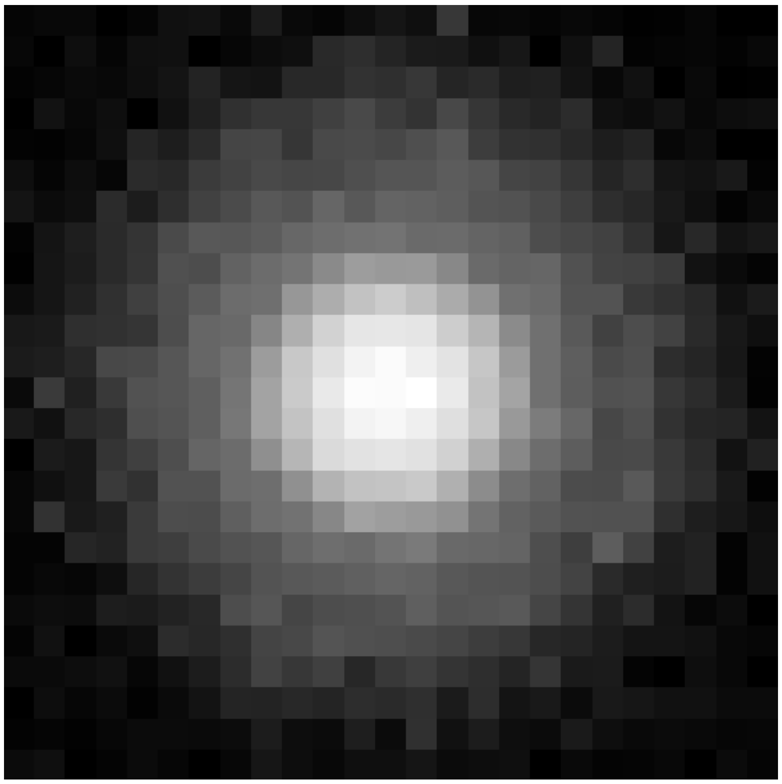


Figure 6.2: Detected beam spread in the empty environment.

In the resulting image in Figure 6.2, one can see the falloff from the center pixel in the received light. While an amount of the emitted light does undergo scattering, a huge majority of the light still reaches the center pixel of the image un-scattered. Thus, in my results, I look for a significant spike in the amount of received light at the center pixel of the image compared to the neighboring pixels. Additionally, one sees the expected falloff in the amount of light received at pixels with distance from the center pixel.

6.3 Rendering Variations of Volume Sphere on Light Field Receiver With FPI-RT

Next, I render a more interesting environment containing a beam emitter casting light onto a light field but with a volume sphere of non-trivial scattering between the beam emitter and camera. This environment is constructed to contain a volume sphere of scattering material embedded in a homogenous environment of homogenous, low-scattering material.

To support arbitrary volume parameters within the environment, I modify my weighting function to, per segment, look up the local scattering parameters at the segment's final position in world space. I then identify the prebaked weighting table with the closest matching scattering parameterization and use that table to calculate the segment's weight. To limit the size of the prebaked lookup table, I allow my environments to contain two scattering configurations: within a high-scattering object (sphere, bunny) and that which is not and is thus a low-scattering environment. When performing the world space lookup of the local scattering parameters, I select the high-scattering object table if the world space position is within the object and the low-scattering environment table when outside for use during segment weighting.

6.3.1 Exploring the Effect of Changing Sphere Size

I first explore the effect of changing the sphere volume geometry on the final image, focusing on the environment shown in Figure 6.3. Note I place the light field within the center of the sphere volume to see the radius of the scattering sphere. I render multiple images, setting the sphere scattering parameter to $\sigma_b = 0.5$ and varying the sphere radius, $r = [2, 3, 5]$. I use paths with 66 segments.

As seen in the resulting images in Figure 6.4, modifying the sphere radius results in the spheres of different radii appearing in the images. Additionally, as the sphere increases in radius, the received light decreases in amplitude as the light transfers/scatters through more volume density. As expected, the received light peaks in the center of the sphere, as most of the emitted light is forward scattered through the sphere. Additionally, the falloff of received light symmetrically falls off from the center of the sphere due to multiple scattering through the outer portions of the sphere.

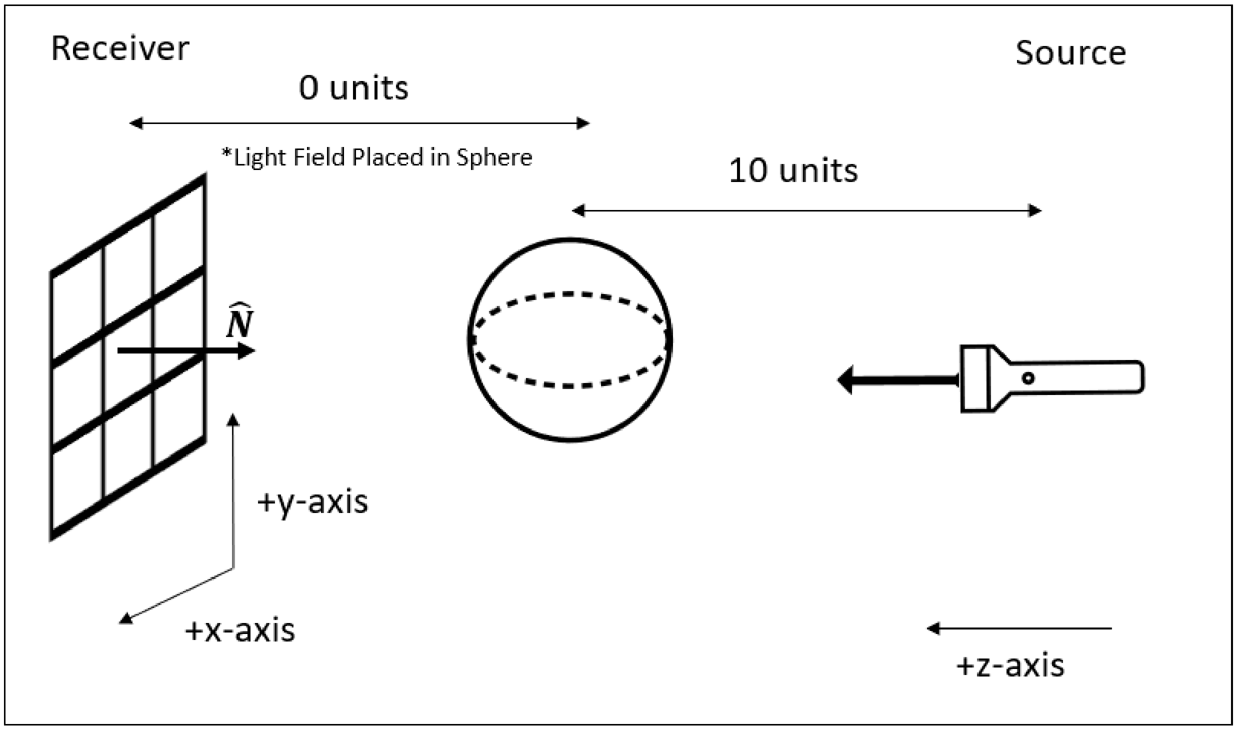
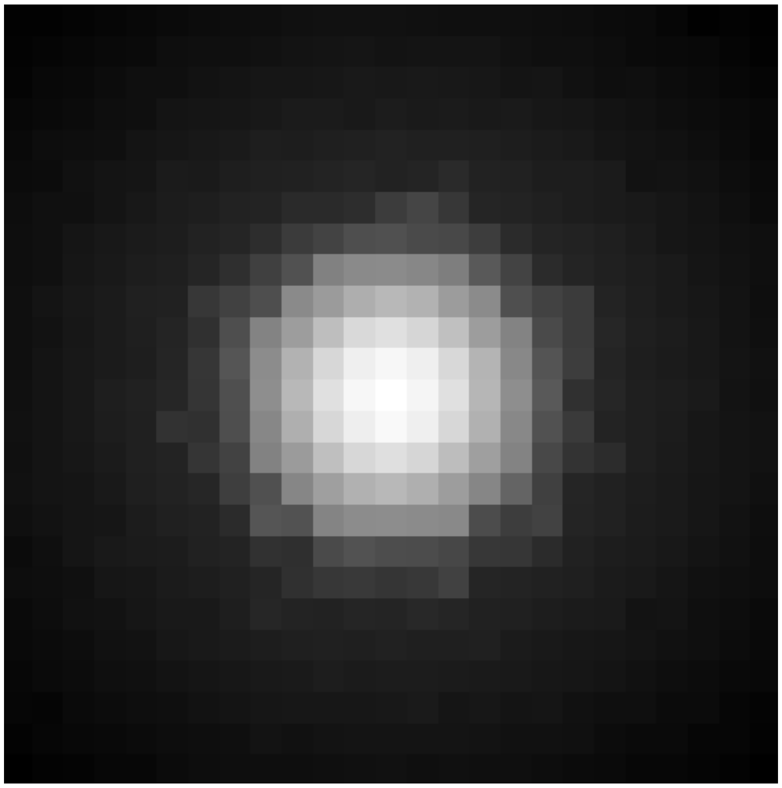
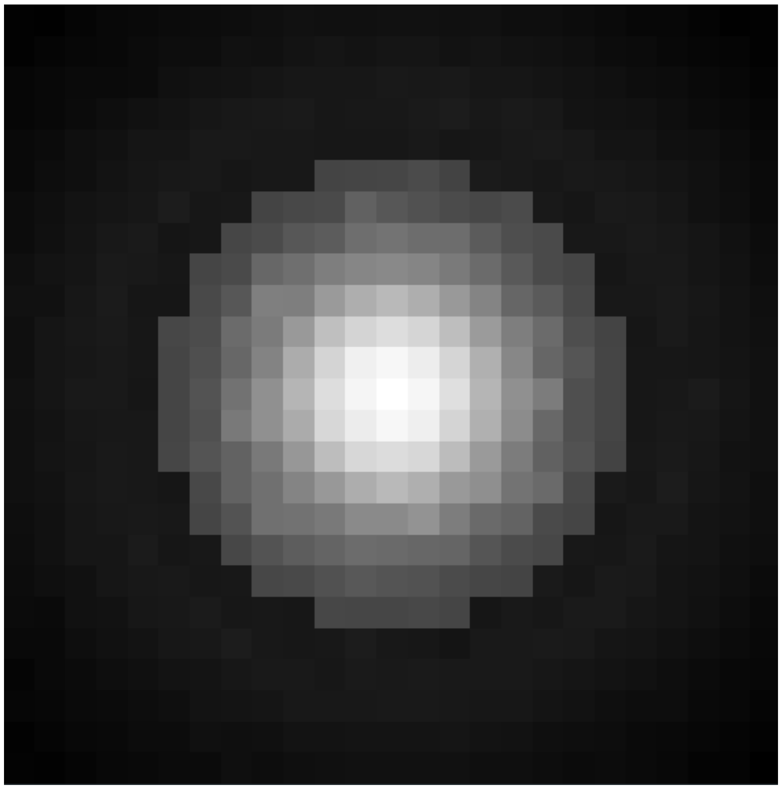


Figure 6.3: Volumetric sphere environment configuration.



(a) Sphere with radius = 2.


(b) Sphere with radius = 3.

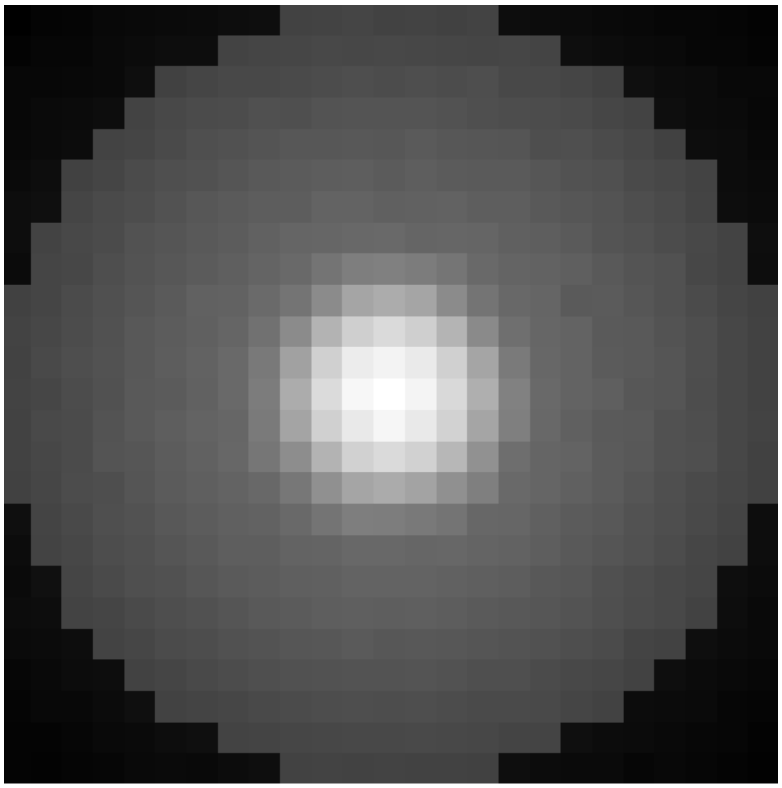

(c) Sphere with radius = 5.

Figure 6.4: Sphere environment with emitter behind sphere.

6.3.2 Exploring the Effect of Changing Emitter Position

A second interesting effect I explore is the effect of changing the emitter position on the final image, with environment construction shown in Figure 6.5. To conduct this, I focus on an environment of a sphere of radius five and a scattering parameter $\sigma_b = 0.5$, varying the emitter location to be placed offset from the center of the sphere in either the x- or y-axis, facing toward the sphere center. Again, paths of 66 segments are used.

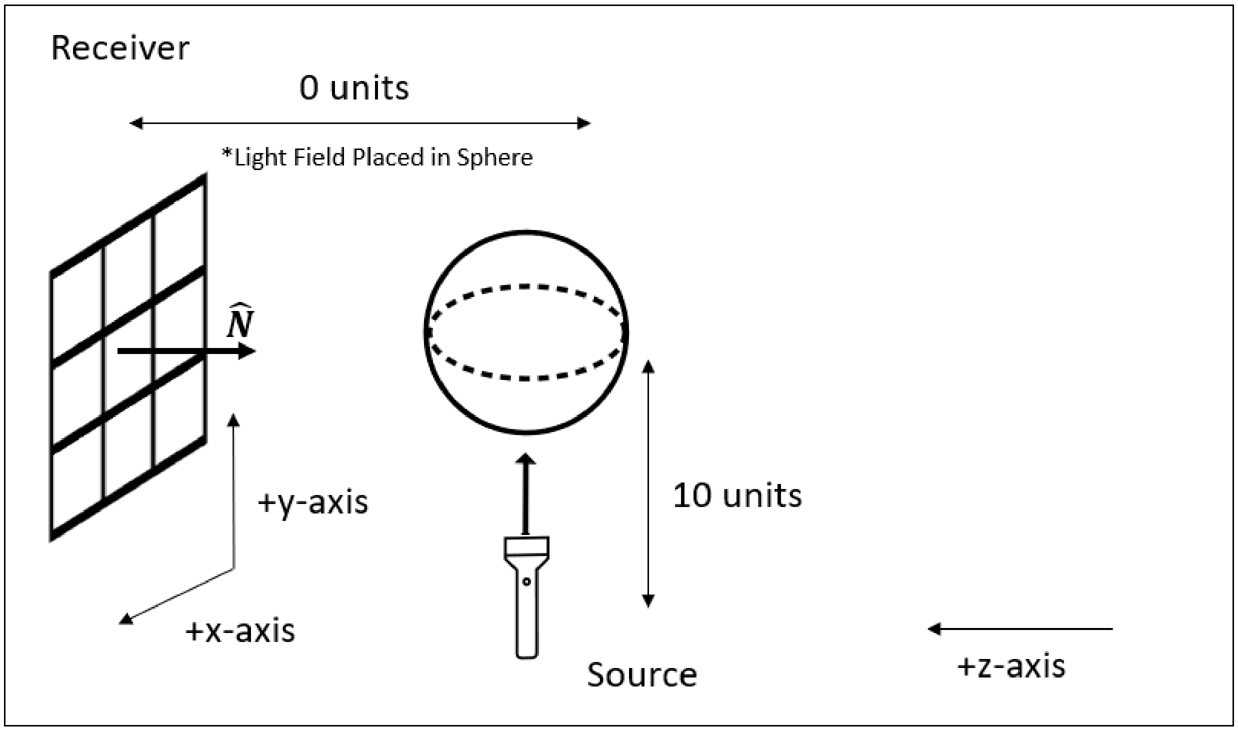
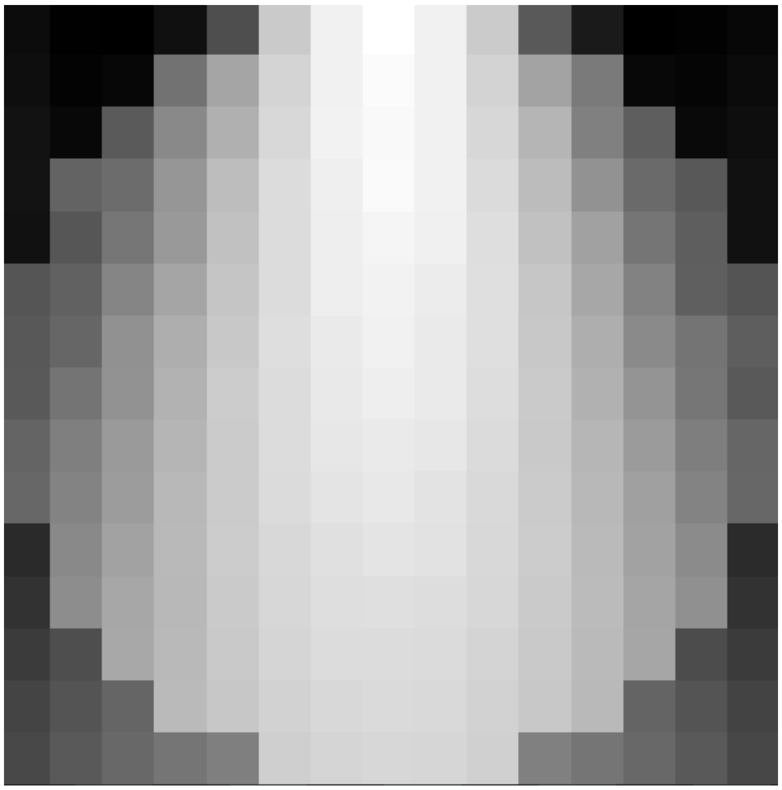
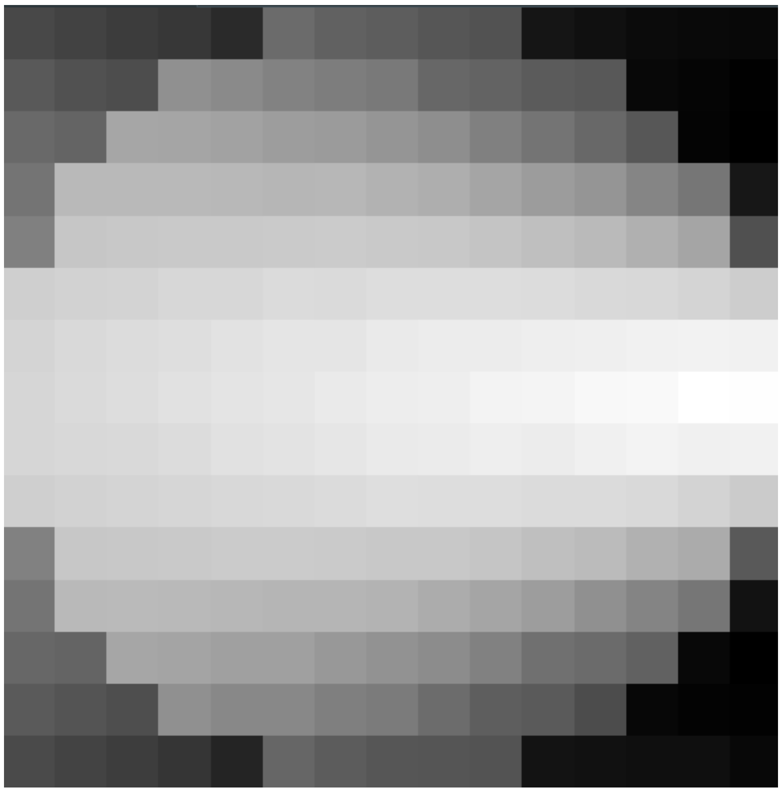
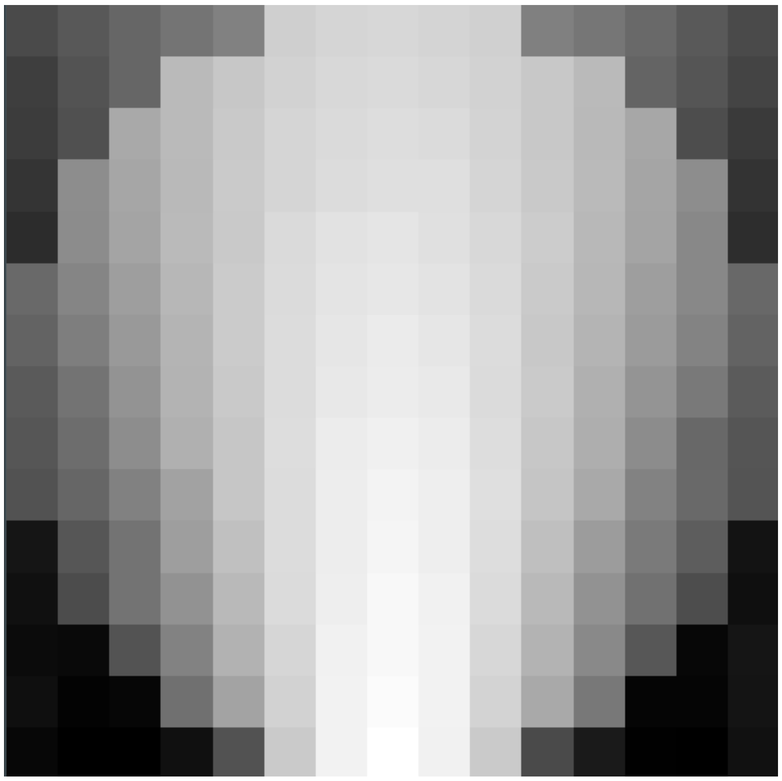


Figure 6.5: Volumetric sphere with side emitter environment configuration.



(a) Sphere with emitter 10 units +y, facing -y.


(b) Sphere with emitter 10 units +x, facing -x.


(c) Sphere with emitter 10 units -y, facing +y.

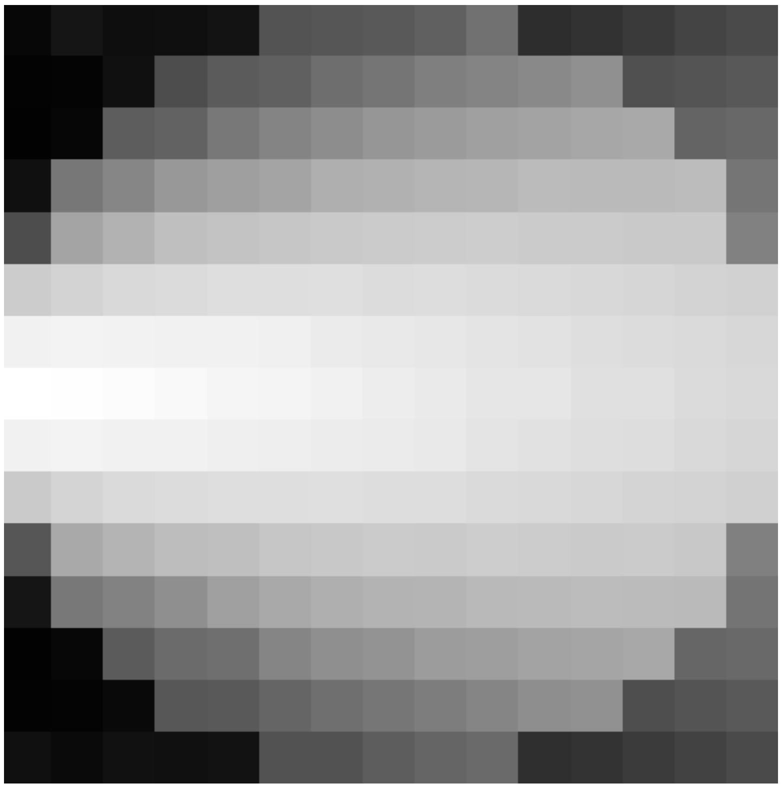
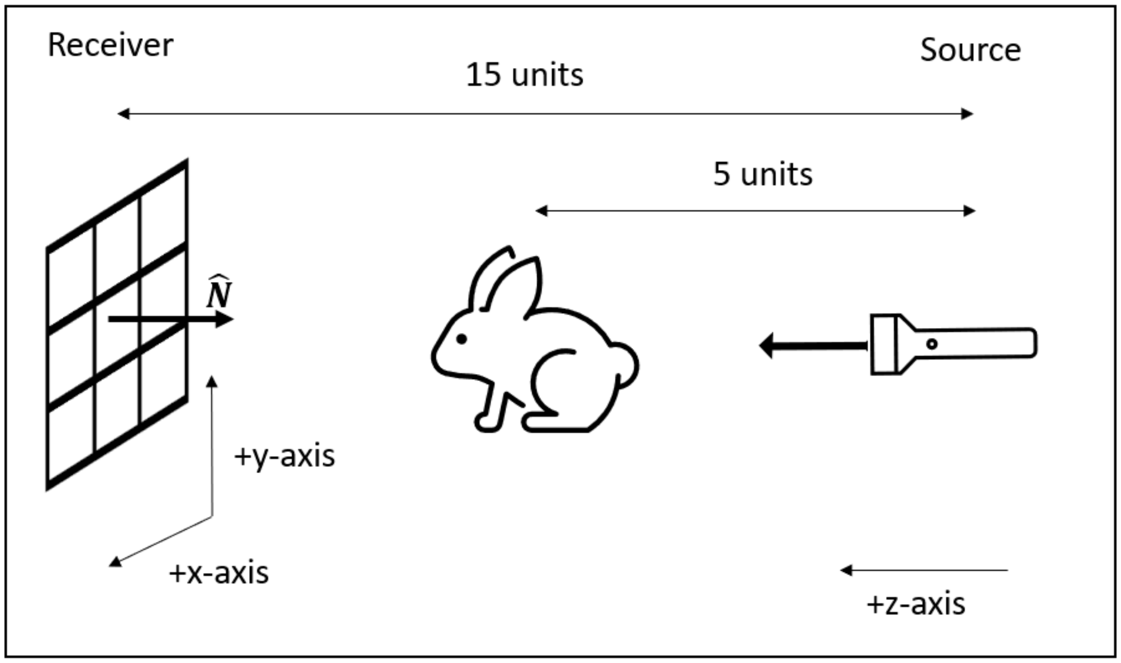

(d) Sphere with emitter 10 units -x, facing +x.

Figure 6.6: Sphere with different emitter directions.

As expected, nearly identical images are rendered for all emitter configurations, with each emitter rotation by 90 degrees around the center z-axis resulting in an equivalent rotation of the resulting image by 90 degrees. Additionally, the side of the sphere struck by the emitter first is the brightest, with the pixels falling off as the light scatters through the sphere volume. Also, note that the environment outside the sphere is darker on the side closest to the emitter, as light has not had a chance to scatter through the volume, while the environment on the far side of the sphere from the emitter is brighter.

6.4 Rendering Complex Volume on Light Field Receiver With FPI-RT

The next experiments I ran involved rendering a complex volume, in my case, a volumetric representation of the Stanford bunny. Like the previous images, I generate this by firing a beam emitter toward an image plane centered such that the beam strikes the exact center of the image. However, this time, I place an OpenVDB volume of the Standford Bunny between the emitter and image plane, as seen in Figure 6.7.



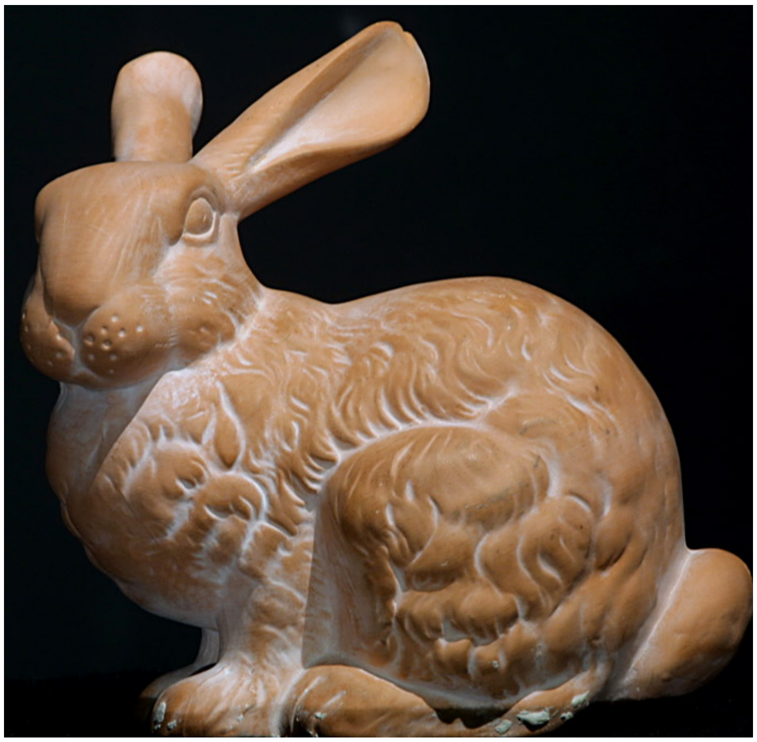
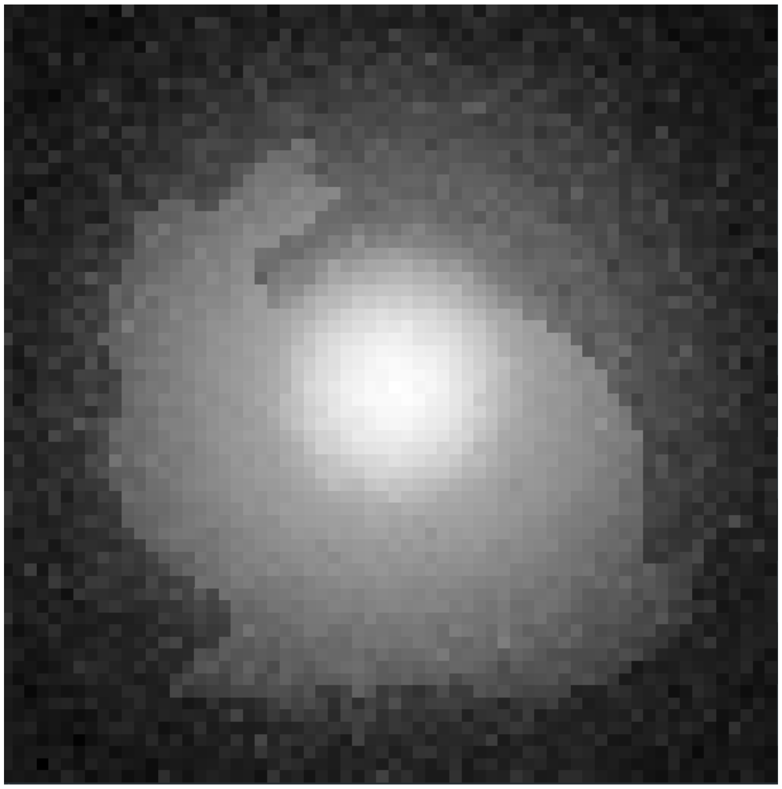
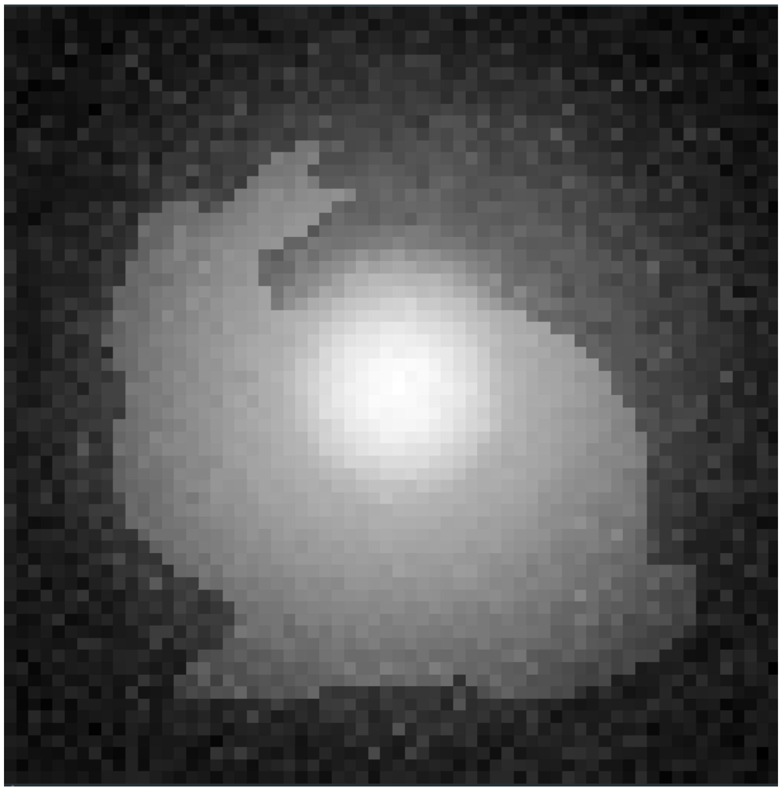


Figure 6.7: Bunny scattering environment configuration. Additionally, a standard visualization of Stanford bunny demonstrates orientation within the environment. This is not rendered using FPI-RT.

First, I explore the behavior of setting the bunny to low scattering coefficients, $\sigma_b = [0.001, 0.01, 0.9]$. The resulting images, shown in Figure 6.8, clearly show the bunny volume, with the majority of received light remaining at the center of the image, as most is forward scattered through the bunny. As the scattering paramter increases, more light is scattered throughout the entirety of the bunny volume, with the received bunny shape increasing in intensity due to this increase in overall scattering throughout the volume.



(a) Volumetric bunny with $\sigma_b = 0.001$.


(b) Volumetric bunny with $\sigma_b = 0.01$.

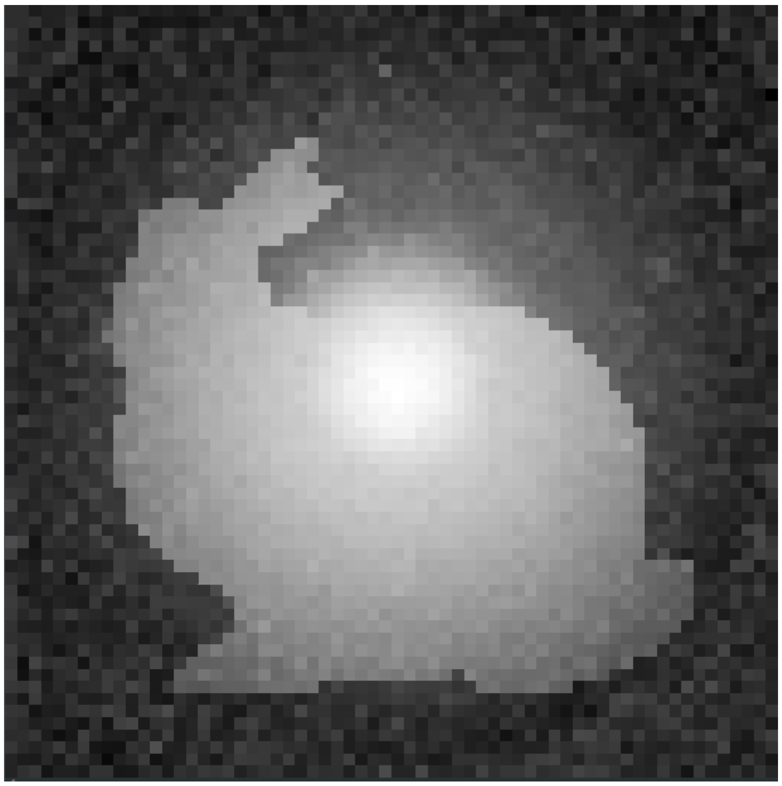
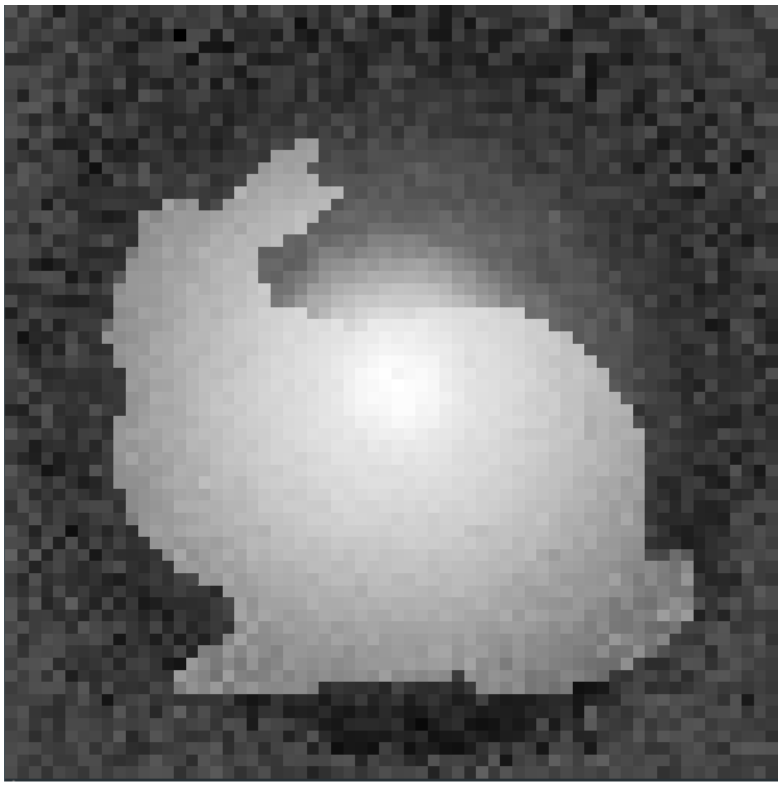
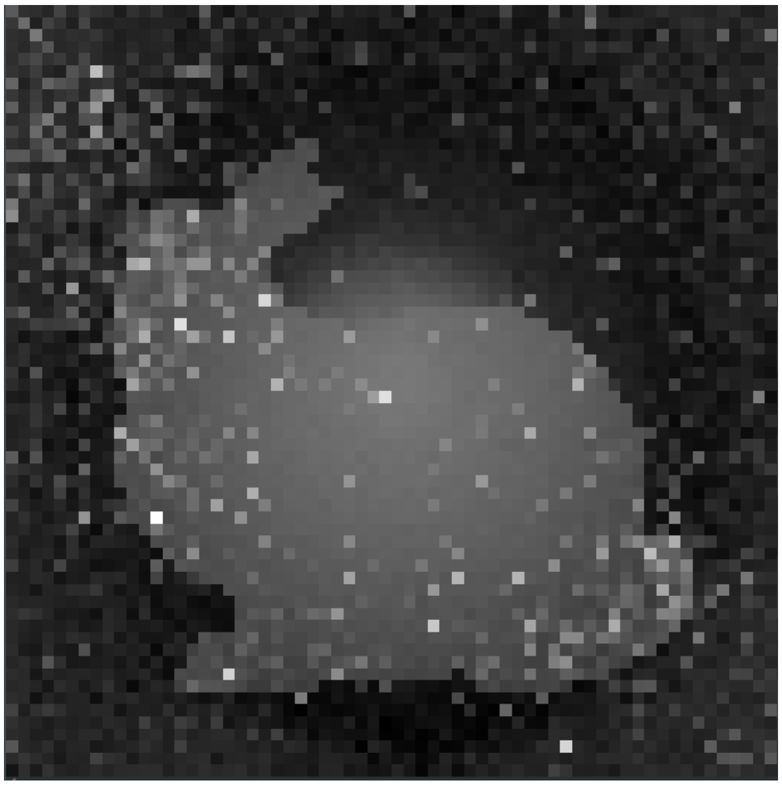
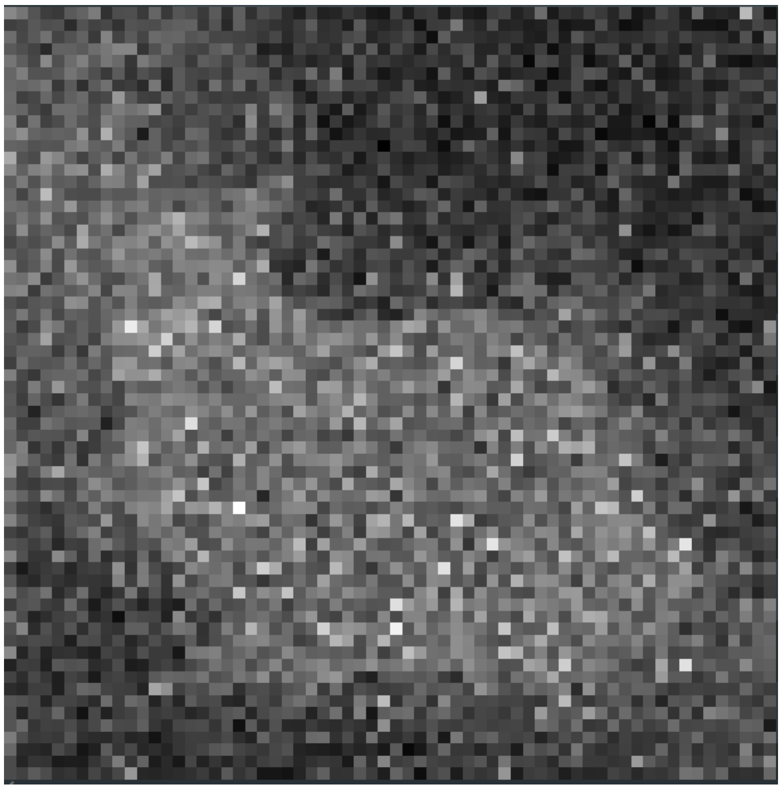

(c) Volumetric bunny with $\sigma_b = 0.9$.

Figure 6.8: Low scattering volumetric bunny environment rendered using FPI-RT path generation solver.

Next, I explore the behavior of setting the bunny to high scattering coefficients, $\sigma_b = [25, 50, 75]$. The resulting images, shown in Figure 6.9, still show the bunny volume. However, as the scattering parameter increases in this high scattering range, the amount of scattering taking place within the bunny leads to a near-uniform scattering within the bunny. Additionally, capturing this near-uniform scattering requires significantly more overall paths, contributing to the increase in noise in the resulting image. Lastly, with an increase in scattering, paths through the bunny become less common, additionally contributing to the increase in noise.


(a) Volumetric bunny with $\sigma_b = 25$.


(b) Volumetric bunny with $\sigma_b = 50$.



(c) Volumetric bunny with $\sigma_b = 75$.

Figure 6.9: High scattering volumetric bunny environment rendered using FPI-RT path generation solver.

6.4.1 Rendering Bunny Lit From Side

Lastly, I explore the effect of lighting the bunny volume from a beam emitter placed directly under the bunny, facing up through the volume, as shown in Figure 6.10. For this, I explore an empty environment of this construction, as well as a range of scattering parameters of the bunny, $\sigma_b = [0.001, 0.01, 0.1, 5]$. The resulting images are shown in Figure 6.11.

As seen in Figure 6.11, the resulting images of a volume bunny lit from below change significantly as the scattering coefficient changes from 0.001 to 5. First, as seen in Figure 6.11a, an empty environment lit from the bottom is dominated by the background scattering of the low, homogenous environment scattering. With introducing a low scattering bunny, $\sigma_b = 0.001$, this environment scattering maintains a dominant role. However, scattering of the bunny model tightens the spread

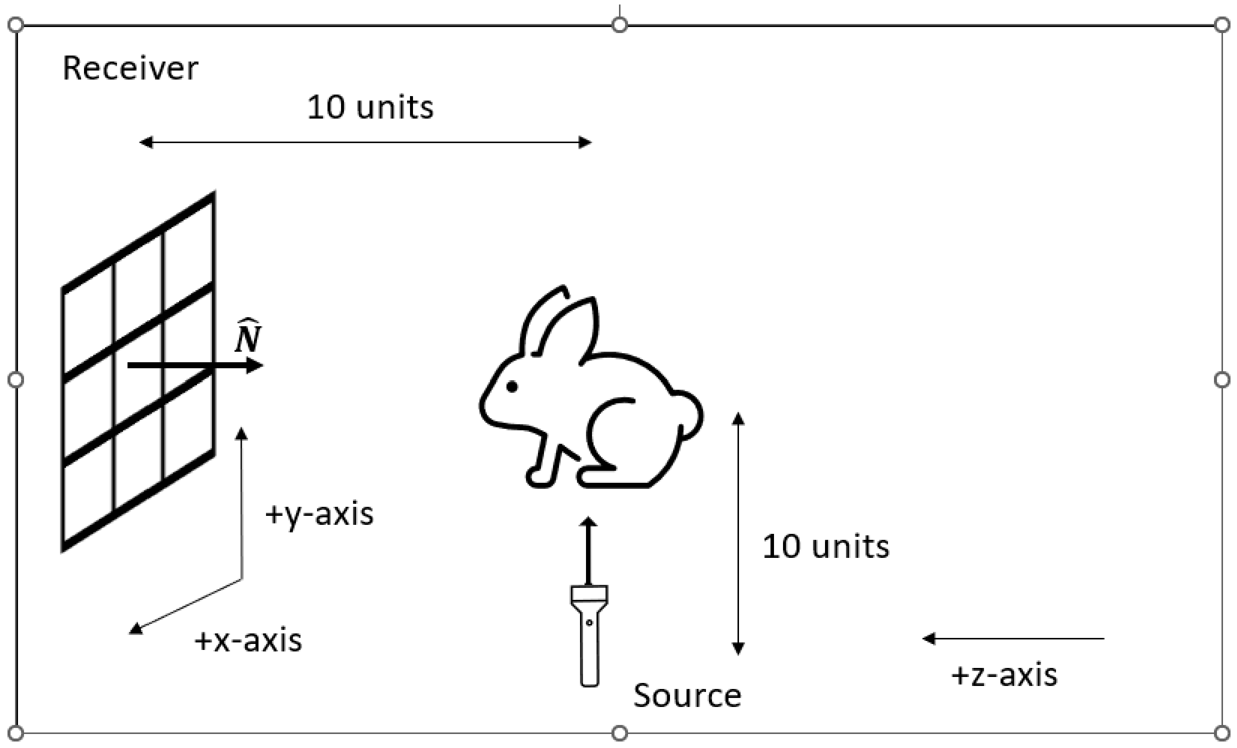
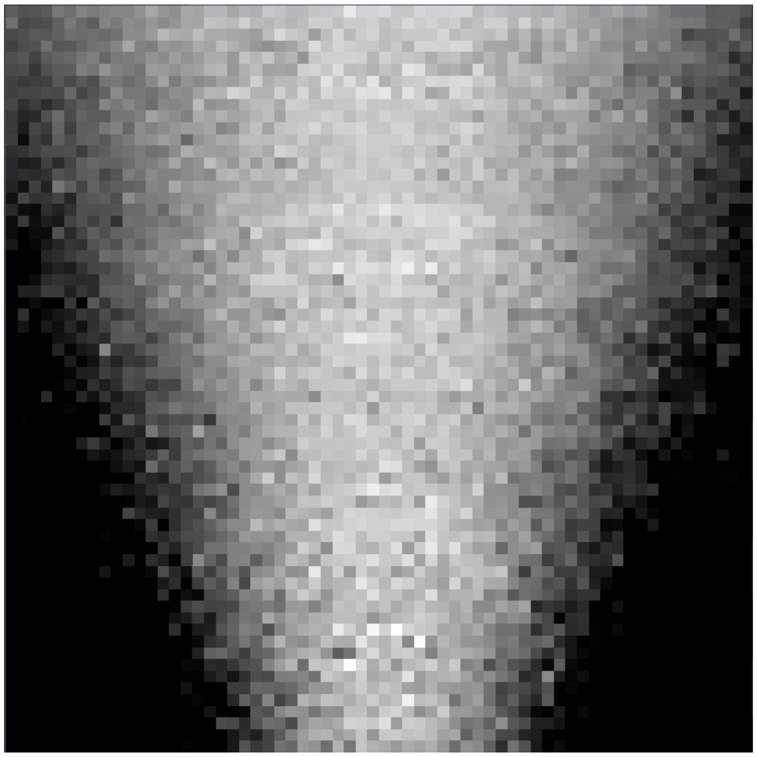
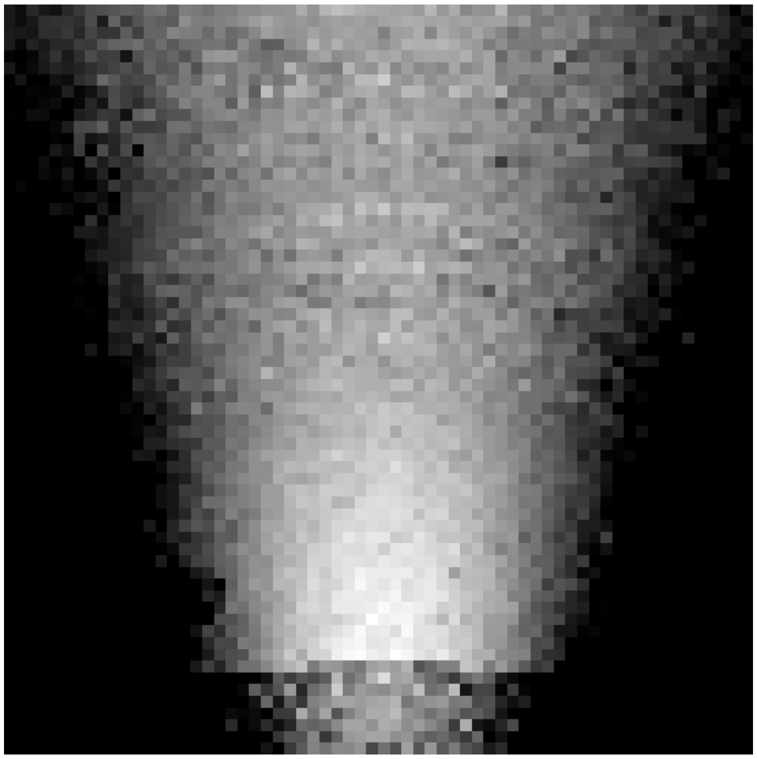
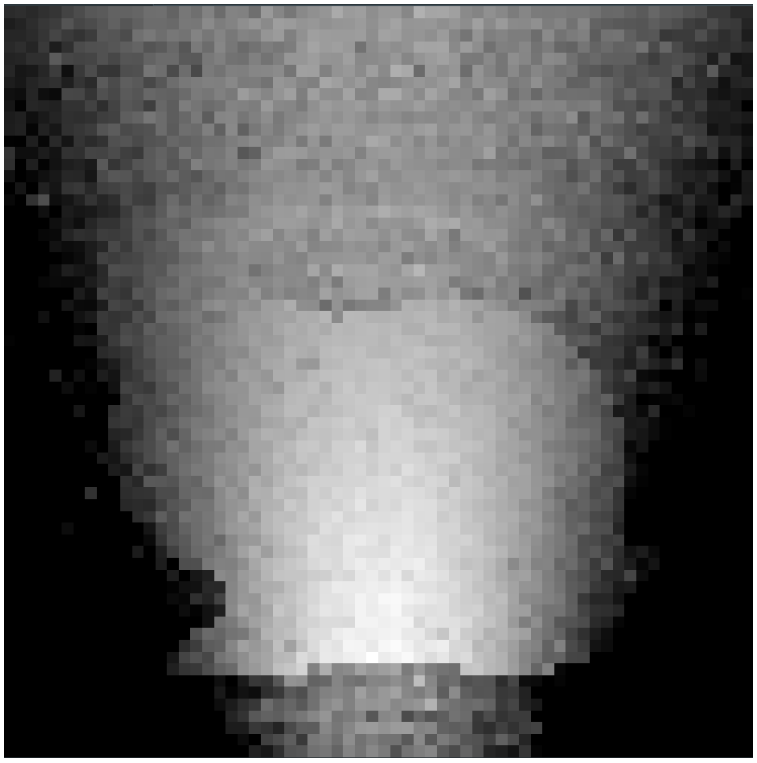


Figure 6.10: Volumetric bunny with side emitter environment configuration.

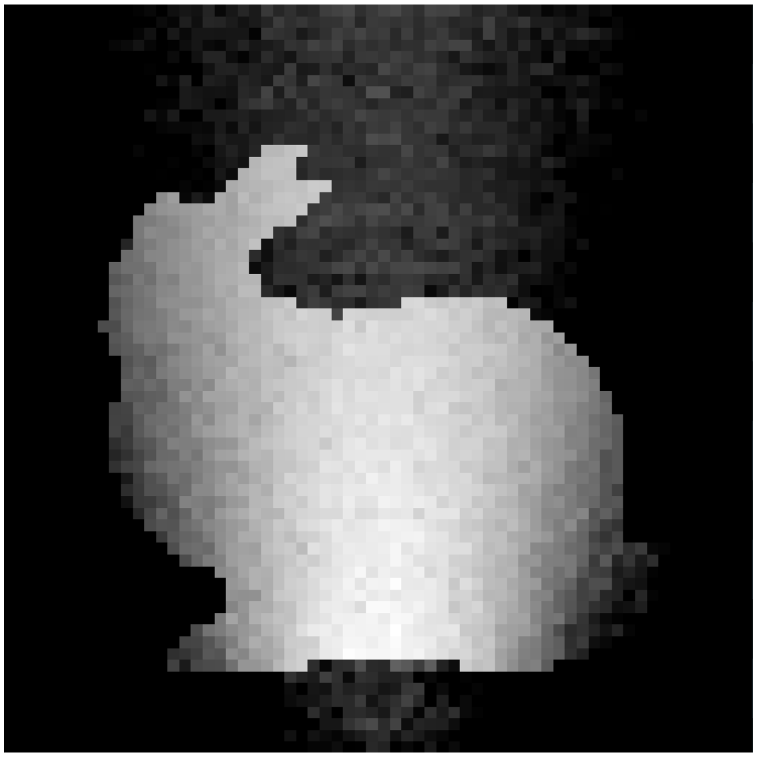
of the entire beam and scatters an amount of light at the base of the bunny, clearly showing the outline of the lowest part of the model. This trend continues as the scattering paramter increases to $\sigma_b = 0.01$, then $\sigma_b = 5$, with the lit bunny becoming clearer as the scattering coefficient increases.


(a) Empty environment.


(b) Volumetric bunny with $\sigma_b = 0.001$.



(c) Volumetric bunny with $\sigma_b = 0.01$.



(d) Volumetric bunny with $\sigma_b = 5$.

Figure 6.11: Volumetric bunny environment rendered using FPI-RT path generation solver, with emitter placed directly under the bunny.

7. CONCLUSION

To conclude, I first summarize the major contributions presented in this work, discussing how all work together to improve computer graphics by improving the computation efficiency of numerically solving the FPI-RT. Then, I discuss several key future directions, which I believe are the next best research directions for expanding the use of the FPI-RT formulation.

7.1 Summary of Contributions

In this work, I presented several contributions which move the computer graphics field closer to utilizing the Feynman Path Integral formulation of radiative transfer when simulating radiative transfer in virtual environments.

1. In a first key contribution, I present and verify a path perturbation numerical solver using a simple yet powerful representation of paths and an efficient path perturbation algorithm for numerically computing the FPI-RT, improving the computation efficiency of the solver by 10^4 times compared to the previous work. The approach is numerically stable and allows efficient CPU and GPU implementation. However, this optimized path-perturbation numerical solver is limited by bias introduced by the path-perturbation approach.

2. In a second key contribution, I present and verify a novel numerical solution approach based on path generation, in which the solver randomly explores valid path space. I validate this solver by comparing it against other numerical solvers for a simplified weighting function, allowing for an analytical derivation of the FPI-SW solution for small segment count paths. Additionally, I demonstrate the convergence of the path generation numerical solver using the full radiative transfer weighting kernel.

3. Third, I demonstrate that the proposed path generation solver is powerful enough to render images within simple environments. While this demonstration falls in the domain of

computer graphics, the application of the solver is not limited to the visible light portion of the electromagnetic spectrum. The method can easily simulate other portions of the spectrum for different research domains, demonstrated by my path perturbation solver first being published in a nuclear engineering journal, the Annals of Nuclear Engineering [36].

All three contributions provide a clear improvement over the previous work, introducing a computationally efficient method of numerically solving the FPI-RT and demonstrating the formulation's applicability to simulating virtual environments. This is a significant step in making the FPI-RT feasible for more general computations.

These advances promise to make FPI-RT feasible for computer graphics, providing a more efficient numerical solution for one of the most powerful formulations of radiative transfer in its generality and ability to capture high scattering orders. With the improvements in computation speed presented in this work and the identification of key directions of future work, I believe the FPI-RT has the potential to be an attractive method to which the field will turn for ground truth comparisons of high orders of scattering in virtual environments.

7.2 Key Directions of Future Work

Working toward utilizing the FPI-FT as a solution for radiative transfer in a virtual environment, I have identified several key follow-up work directions.

One key direction of future work will be to fully explore and formalize the pdfs of samples within valid path space. The first key pdf that needs to be formalized is the pdf relating angle parameterization during path generation to the distribution of samples in path space. This will correct any bias that might exist in the path generation and angle integration numerical solvers. Additionally, a second key pdf relates the selected arclength of a path to the distribution of samples in the full valid path space created by integrating over a continuous range of arclengths. Including correct handling of this arclength pdf will remove any bias in the integration over the FPI-RT during environment renders.

A second promising direction of work is to explore the use of importance sampling during an FPI-RT and when integrating over the FPI-RT for a full environment solution. While my proposed numerical solver is very efficient, especially compared to the prior numerical solver, the evaluation of the FPI-RT is still too slow for full environment renders. If performed correctly, importance sampling can significantly increase the convergence rate of a single FPI-RT computation and thus could decrease the computation time of performing an environment render significantly. An obvious target distribution for importance sampling would be to sample the path weighting function directly. However, as this path weighting function is very complex, I believe a different strategy should be utilized. Rather than sampling directly from a pdf proportional to the path weighting function, samples could be drawn from the angle parameterization and arclength pdfs proposed as the first future work. These samples could then be utilized directly using standard importance sampling or reweighted using re-weighted importance sampling to better target the FPI-RT samples to the target path weighting pdf.

Another key direction of future work will involve fully deriving and computing the normalizing term for the full path integral solution, $\mathbb{D}$. While in this work, I have presented a method of numerically computing the FPI-RT and was able to apply the solver to simulate the transfer of light through a virtual environment, the generated results are not physically accurate as they lack this key normalization. For example, one manifestation of this lack of normalizer is the inability to directly compare the results of FPI-RT calculations performed with different segment counts. Thus, to present the images seen in this work and compare the results of multiple numerical solvers of the FPI-RT for paths of the same number of segments, it was necessary to compare only the relative behavior of the results rather than the exact numerical values. Additionally, the lack of a normalizer prevents combining the results of multiple FPI-RT solutions for paths of different numbers of segments. This approach could potentially increase the method's performance by selecting the minimally required number of segments depending on the environment parameterization and region of valid path space.

A major challenge of computing this normalizer is that as the number of segments in a path increases, the normalizing component regarding the number of path segments becomes increasingly complex and computationally expensive. This normalizing term, derived in Appendix B, is recursive in construction, exponentially increasing computation time with the number of segments in a path. I can compute this normalizing term with current hardware for up to six segment paths. However, any normalization terms for a greater number of segments remain computationally preventative. Additionally, the full FPI-RT normalizer must include correct handling of the path generation sampling pdf.

While these future steps will take significant work, they should enable FPI-RT to be a fully usable rendering mechanism for computer graphics and other domains.

REFERENCES

[1] J. F. Blinn, “Light reflection functions for simulation of clouds and dusty surfaces,” *ACM SIGGRAPH Computer Graphics*, vol. 16, pp. 21–29, jul 1982.

[2] J. T. Kajiya, “The rendering equation,” *ACM SIGGRAPH Computer Graphics*, vol. 20, pp. 143–150, aug 1986.

[3] A. Schuster, “Radiation through a foggy atmosphere,” *The Astrophysical Journal*, vol. 21, p. 1, jan 1905.

[4] S. Chandrasekhar, *Radiative Transfer.* Dover Publications, 1960.

[5] E. Lommel, “Die photometrie der diffusen zurückwerfung,” *Annalen der Physik*, vol. 272, no. 2, pp. 473–502, 1889.

[6] J. Arvo, “Transfer equations in global illumination,” *Global Illumination, SIGGRAPH ’93 Course Notes*, vol. 42, 1993.

[7] J. T. Kajiya and B. P. V. Herzen, “Ray tracing volume densities,” *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 165–174, jul 1984.

[8] E. P. Lafortune and Y. D. Willems, “Rendering participating media with bidirectional path tracing,” in *Eurographics*, pp. 91–100, Springer Vienna, 1996.

[9] M. Pauly, T. Kollig, and A. Keller, “Metropolis light transport for participating media,” in *Eurographics*, pp. 11–22, Springer Vienna, 2000.

[10] H. W. Jensen and P. H. Christensen, “Efficient simulation of light transport in scences with participating media using photon maps,” in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, ACM Press, 1998.

[11] S. Premoze, M. Ashikhmin, J. Tessendorf, R. Ramamoorthi, and S. Nayar, "Practical rendering of multiple scattering effects in participating media," 2004.

[12] J. Tessendorf, "Radiative transfer as a sum over paths," *Physical Review A*, vol. 35, pp. 872–878, jan 1987.

[13] P. Kilgo, *Radiative Transfer Using Path Integrals for Multiple Scattering in Participating Media*. PhD thesis, Clemson University, 2016.

[14] J. Tessendorf, "Numerical integration of the feynman path integral for radiative transport," *American Nuclear Society*, 2009.

[15] D. S. Ebert and R. E. Parent, "Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques," *ACM SIGGRAPH Computer Graphics*, vol. 24, pp. 357–366, sep 1990.

[16] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion, "A survey on participating media rendering techniques," *The Visual Computer*, vol. 21, pp. 303–328, jun 2005.

[17] J. Křivánek, I. Georgiev, T. Hachisuka, P. Vévoda, M. Šik, D. Nowrouzezahrai, and W. Jarosz, "Unifying points, beams, and paths in volumetric light transport simulation," *ACM Transactions on Graphics*, vol. 33, pp. 1–13, jul 2014.

[18] B. Bitterli and W. Jarosz, "Beyond points and beams," *ACM Transactions on Graphics*, vol. 36, pp. 1–12, jul 2017.

[19] J. Stam, "Multiple scattering as a diffusion process," in *Eurographics*, pp. 41–50, Springer Vienna, 1995.

[20] E. D'Eon and G. Irving, "A quantized-diffusion model for rendering translucent materials," *ACM Transactions on Graphics*, vol. 30, pp. 1–14, jul 2011.

[21] J. R. Frisvad, T. Hachisuka, and T. K. Kjeldsen, "Directional dipole model for subsurface scattering," *ACM Transactions on Graphics*, vol. 34, pp. 1–12, dec 2014.

[22] L. Szirmay-Kalos, M. Sbert, and T. Ummenhoffer, "Real-time multiple scattering in participating media with illumination networks," 2005.

[23] R. Lee and C. O'Sullivan, "Accelerated light propagation through participating media," 2007.

[24] T. Müller, M. Papas, M. Gross, W. Jarosz, and J. Novák, "Efficient rendering of heterogeneous polydisperse granular media," *ACM Transactions on Graphics*, vol. 35, pp. 1–14, nov 2016.

[25] S. Kallweit, T. Müller, B. McWilliams, M. Gross, and J. Novák, "Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks," Sept. 2017.

[26] R. P. Feynman, *Quantum mechanics and path integrals*. Dover Publications, 2010.

[27] J. Tessendorf, "Time-dependent radiative transfer and pulse evolution," *Journal of the Optical Society of America A*, vol. 6, p. 280, feb 1989.

[28] P. Kilgo and J. Tessendorf, "Beam spread functions calculated using feynman path integrals," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 196, pp. 149–154, jul 2017.

[29] J. Petržala, "Revision of path-integral approach to radiative transfer," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 270, p. 107670, aug 2021.

[30] L. T. Perelman, J. Wu, I. Itzkan, and M. S. Feld, "Photon migration in turbid media using path integrals," *Physical Review Letters*, vol. 72, pp. 1341–1344, feb 1994.

[31] L. T. Perelman, J. Wu, Y. Wang, I. Itzkan, R. R. Dasari, and M. S. Feld, "Time-dependent photon migration using path integrals," *Physical Review E*, vol. 51, pp. 6134–6141, jun 1995.

[32] M. J. Wilson and R. K. Wang, “A path-integral model of light scattered by turbid media,” *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 34, pp. 1453–1472, apr 2001.

[33] S. Premoze, M. Ashikhmin, and P. Shirley, “Path integration for light transport in volumes,” 2003.

[34] W. Jarosz, *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, 2008.

[35] E. Kreyszig, *Differential geometry*. Dover Publications, 1991.

[36] B. Taylor, J. Keyser, and J. Tessendorf, “Path integral radiative transfer via polyline representation allowing GPU implementation,” *Annals of Nuclear Energy*, vol. 173, p. 109098, aug 2022.

APPENDIX A

DERIVATION OF FEYNMAN PATH INTEGRAL FORMULATION OF RADIATIVE TRANSFER

I begin with the integral form of the Rendering Transport Equation.

$$L(\vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R) = \int_0^\infty ds \int d^3\vec{\boldsymbol{x}}_S \int_{4\pi} d\Omega \, G(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S) \, Q(\vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S) \tag{A.1}$$

In Chapter 3, I describe that the FPI formulation of radiative transfer utilizes the following transport kernel.

$$\begin{aligned} G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = e^{-\sigma_t \Delta s} \int D\mathbb{P} Dp \, \delta\left(\hat{\boldsymbol{\beta}}\left(0\right) - \hat{\boldsymbol{\omega}}_S\right) \, \delta\left(\hat{\boldsymbol{\beta}}\left(S\right) - \hat{\boldsymbol{\omega}}_R\right) \\ \times \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right) \, \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R\right) \\ \times \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S ds' \hat{\boldsymbol{\beta}}\left(s'\right)\right) \\ \times \exp\left(\sigma_b \int_0^S ds' \, \widetilde{\boldsymbol{Z}}\left(\vec{\boldsymbol{p}}\left(s'\right)\right)\right) \\ \times \exp\left(i \int_0^S ds' \, \vec{\boldsymbol{p}}\left(s'\right) \cdot \frac{d\hat{\boldsymbol{\beta}}\left(s'\right)}{ds'}\right) \end{aligned} \tag{A.2}$$

A.1 Deriving the Path Integral Formulation of Radiative Transfer

I start with a discussion of what is represented by this transport kernel. The kernel, $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$ represents the amount of light that travels from a starting position $\vec{\boldsymbol{x}}_S$ and direction $\hat{\boldsymbol{\omega}}_S$ to an ending position $\vec{\boldsymbol{x}}_R$ and direction $\hat{\boldsymbol{\omega}}_R$ in time S. One useful property of this transport kernel allows the propagator to be written as the combined effects of two propagators, combined via integration over a linking spatial position $\vec{\boldsymbol{x}}''$ and direction $\hat{\boldsymbol{\omega}}''$.

$$G\left(S,\vec{\boldsymbol{x}}_R,\hat{\boldsymbol{\omega}}_R,\vec{\boldsymbol{x}}_S,\hat{\boldsymbol{\omega}}_S\right)=\int d^3\vec{\boldsymbol{x}}''\int d^2\,\hat{\boldsymbol{\omega}}''G\left(S',\vec{\boldsymbol{x}}_R,\hat{\boldsymbol{\omega}}_R,\vec{\boldsymbol{x}}'',\hat{\boldsymbol{\omega}}''\right)\,G\left(S-S',\vec{\boldsymbol{x}}'',\hat{\boldsymbol{\omega}}'',\vec{\boldsymbol{x}}_S,\hat{\boldsymbol{\omega}}_S\right) \tag{A.3}$$

With this important property defined, I now discretize the transport kernel into M segments, covering equal time $\Delta s=\frac{S}{M}$. I maintain the boundary conditions by assuming $\vec{\boldsymbol{x}}_M=\vec{\boldsymbol{x}}_R$, $\vec{\boldsymbol{x}}_0=\vec{\boldsymbol{x}}_S$, $\hat{\boldsymbol{\beta}}_M=\hat{\boldsymbol{\omega}}_R$ and $\hat{\boldsymbol{\beta}}_0=\hat{\boldsymbol{\omega}}_S$.

$$G\left(S,\vec{\boldsymbol{x}}_R,\hat{\boldsymbol{\omega}}_R,\vec{\boldsymbol{x}}_S,\hat{\boldsymbol{\omega}}_S\right)=\int d^3\vec{\boldsymbol{x}}_i\int d^2\hat{\omega}_i\prod_{i=1}^{M}G\left(\Delta s,\vec{\boldsymbol{x}}_i,\hat{\omega}_i,\vec{\boldsymbol{x}}_{i-1},\hat{\omega}_{i-1}\right) \tag{A.4}$$

Next, simply taking the limit of $M\rightarrow\infty$, and thus $\Delta s\rightarrow 0$, I have that each small Δs length interval can be expressed as a single scattering event. In doing so, I logically assume that no absorption or scattering is performed along the length of the actual segment of length Δs and instead model the absorption or scattering at the end of the segment, i.e., the connection point between neighboring segments. Thus, I obtain the following given an absorption paramter σ_a and scattering paramter σ_b.

$$\begin{aligned}G\left(\Delta s,\vec{\boldsymbol{x}}_i,\hat{\omega}_i,\vec{\boldsymbol{x}}_{i-1},\hat{\omega}_{i-1}\right)&=(1-\sigma_a\Delta s)\delta(\vec{\boldsymbol{x}}_i-\vec{\boldsymbol{x}}_{i-1}-\hat{\omega}_i\Delta s)\\&\times\;((1-\sigma_b\Delta s)\delta(\hat{\omega}_i-\hat{\omega}_{i-1})+\sigma_b\Delta s\;P(\hat{\omega}_i,\hat{\omega}_{i-1}))\end{aligned} \tag{A.5}$$

Plugging this into the full expression, I obtain the following. Note, I simplify the absorption term as I assume $M\rightarrow\infty$.

$$
\begin{aligned}
G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) &= \int \prod_{i=1}^{M-1} d^3\vec{\boldsymbol{x}}_i \; d^2\hat{\omega}_i (1-\sigma_a \Delta s)^M \\
&\times \prod_{i=1}^{M} \delta(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_{i-1} - \hat{\omega}_i \Delta s) \\
&\times \prod_{i=1}^{M} \left((1-\sigma_b \Delta s)\delta(\hat{\omega}_i - \hat{\omega}_{i-1}) + \sigma_b \Delta s \; P(\hat{\omega}_i, \hat{\omega}_{i-1})\right) \\
G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) &= e^{-\sigma_a \Delta s} \int \prod_{i=1}^{M-1} d^3\vec{\boldsymbol{x}}_i \; d^2\hat{\omega}_i \\
&\times \prod_{i=1}^{M} \delta(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_{i-1} - \hat{\omega}_i \Delta s) \\
&\times \prod_{i=1}^{M} \left((1-\sigma_b \Delta s)\delta(\hat{\omega}_i - \hat{\omega}_{i-1}) + \sigma_b \Delta s \; P(\hat{\omega}_i, \hat{\omega}_{i-1})\right) \qquad \text{(A.6)}
\end{aligned}
$$

I continue by further assuming that I have a phase function that depends only on the angle between the incoming and outgoing directions $P(\hat{\omega}_i - \hat{\omega}_{i-1})$. In doing so, I can then represent the phase function using its Fourier representation. In this representation, $\widetilde{\boldsymbol{Z}}$ represents the Fourier representation of the phase function P, with $\vec{\boldsymbol{p}}$ representing the three-dimensional Fourier integration variable.

$$
P(\hat{\omega}_i - \hat{\omega}_{i-1}) = \frac{1}{(2\pi)^3} \int d^3\vec{\boldsymbol{p}} \widetilde{\boldsymbol{Z}}(\vec{\boldsymbol{p}}) \; exp\left(i\vec{\boldsymbol{p}} \cdot (\hat{\omega}_i - \hat{\omega}_{i-1})\right) \qquad \text{(A.7)}
$$

Likewise, I also make use of the following definition.

$$
\delta(\vec{\boldsymbol{x}}) = \frac{1}{(2\pi)^3} \int d^3\vec{\boldsymbol{p}} \, exp\left(i\vec{\boldsymbol{p}} \cdot \vec{\boldsymbol{x}}\right) \qquad \text{(A.8)}
$$

Plugging these into the full representation, I obtain the following.

$$
\begin{aligned}
G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = e^{-\sigma_t \Delta s} \int \prod_{i=1}^{M-1} d^3\vec{\boldsymbol{x}}_i \; d^2\hat{\omega}_i \\
\times \prod_{i=1}^{M} \delta(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_{i-1} - \hat{\omega}_i \Delta s) \\
\times \prod_{i=1}^{M} \frac{d^3\vec{\boldsymbol{p}}_j}{(2\pi)^3} \, exp(\sigma_b \sum_{i=1}^{M} \Delta s \; \widetilde{\boldsymbol{Z}}(\vec{\boldsymbol{p}}_j)) \\
\times \, exp(i \sum_{i=1}^{M} \vec{\boldsymbol{p}}_j \cdot (\hat{\omega}_i - \hat{\omega}_{i-1}))
\end{aligned}
\tag{A.9}
$$

Now, I take the limit of $M \to \infty$ to make a few substitutions. First, direction vectors $\hat{\omega}$ are now written as a tangent vector of length Δs along the curve, $\vec{\boldsymbol{x}}_j = \hat{\boldsymbol{\beta}}(j\Delta s)$. Additionally, I rewrite $\sum \Delta s$ as $\int ds$, and $\hat{\omega}_i - \hat{\omega}_{i-1}$ as $\frac{d\hat{\boldsymbol{\beta}}(s)}{\Delta s}$. Lastly, I provide two delta functions that enforce the boundary condition directions in $\mathbb{R}$.

$$
\begin{aligned}
G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = e^{-\sigma_t \Delta s} \int D\mathbb{P}Dp \, \delta\left(\hat{\boldsymbol{\beta}}\left(0\right) - \hat{\boldsymbol{\omega}}_S\right) \; \delta\left(\hat{\boldsymbol{\beta}}\left(S\right) - \hat{\boldsymbol{\omega}}_R\right) \\
\times \, \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right) \; \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R\right) \\
\times \, \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S ds' \hat{\boldsymbol{\beta}}\left(s'\right)\right) \\
\times \exp\left(\sigma_b \int_0^S ds' \; \widetilde{\boldsymbol{Z}}\left(\vec{\boldsymbol{p}}\left(s'\right)\right)\right) \\
\times \exp\left(i \int_0^S ds' \, \vec{\boldsymbol{p}}\left(s'\right) \cdot \frac{d\hat{\boldsymbol{\beta}}\left(s'\right)}{ds'}\right)
\end{aligned}
\tag{A.10}
$$

Thus, I have fully derived the propagation kernel.

A.2 Selection of Phase Function

Next, now that I have the path integral form, I need to tease out the weighting for a single path.

I select the Fourier phase function, $\widetilde{\boldsymbol{Z}}(\hat{\boldsymbol{\omega}}_R - \hat{\boldsymbol{\omega}}_S)$, utilized in prior work [14, 13]. Note, this function only depends on the magnitude difference between the incoming and outgoing directions $\hat{\boldsymbol{\omega}}_S$ and $\hat{\boldsymbol{\omega}}_R$, and thus I can write the phase function as taking this single magnitude as input.

$$\widetilde{\boldsymbol{Z}}(p) = N_p \, exp(\frac{-\mu p^2}{2}) \tag{A.11}$$

$$N_p = \frac{\sqrt{\pi\mu/2}}{1 - exp(-2/\mu)} \tag{A.12}$$

A.3 Deriving Single Path Weight

Last, I finish deriving a single path's weighting function $\mathbb{W}(\mathbb{P})$.

I begin with the derived path integral, Equation A.10. I remove any integration over paths $\int D\mathbb{P}$ and thus focus only on a single, valid path $\mathbb{P}$. Note that I do not write the valid path enforcing delta functions due to this validity assumption.

$$\begin{aligned} \mathbb{W}(\mathbb{P}) = e^{-\sigma_t S} \frac{d^3\vec{\boldsymbol{p}}}{(2\pi)^3} \exp\left(\sigma_b \int_0^S ds' \, \widetilde{\boldsymbol{Z}}\left(\vec{\boldsymbol{p}}(s')\right)\right) \\ \times \exp\left(i \int_0^S ds' \, \vec{\boldsymbol{p}}(s') \cdot \frac{d\hat{\boldsymbol{\beta}}(s')}{ds'}\right) \end{aligned} \tag{A.13}$$

Now, I can rewrite this in a discrete form by taking $\mathbb{P}$ as a path of M segments. Note, I rewrite $\frac{d\hat{\boldsymbol{\beta}}(s')}{ds'}$ as $\hat{N}_i \kappa_i$, representing the local curvature of segment n, κ_n, which curves in direction $\hat{N}_i$.

$$\mathbb{W}(\mathbb{P}) = \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \frac{d^3\vec{\boldsymbol{p}}}{(2\pi)^3} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(|\vec{\boldsymbol{p}}|\right) + i\vec{\boldsymbol{p}} \cdot \hat{N}_i \kappa_i \Delta s\right) \tag{A.14}$$

I can now rewrite this equation with $\vec{\boldsymbol{p}}$ in spherical coordinates.

$$\begin{aligned}\mathbb{W}(\mathbb{P}) &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{2\pi} d\phi \int_0^{\infty} \int_0^{\pi} d\theta dp \, \frac{p^2 \, sin(\theta)}{(2\pi)^3} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(|\vec{\boldsymbol{p}}|\right) + i\vec{\boldsymbol{p}} \cdot \hat{N}_i \kappa_i \Delta s\right) \\ &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{\infty} \int_0^{\pi} d\theta dp \, \frac{p^2 \, sin(\theta)}{(2\pi)^2} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(|\vec{\boldsymbol{p}}|\right) + i\vec{\boldsymbol{p}} \cdot \hat{N}_i \kappa_i \Delta s\right)\end{aligned} \tag{A.15}$$

Next, I continue by rewriting $\vec{\boldsymbol{p}} \cdot \hat{N}_n$ as $pcos(\theta)$.

$$\begin{aligned}\mathbb{W}(\mathbb{P}) &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{\infty} \int_0^{\pi} d\theta dp \, \frac{p^2 \, sin(\theta)}{(2\pi)^2} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(|\vec{\boldsymbol{p}}|\right) + i \, p \, cos(\theta) \kappa_i \Delta s\right) \\ &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{\infty} dp \, \frac{p^2}{(2\pi)^2} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(p\right)\right) \\ &\times \int_0^{\pi} d\theta sin(\theta) \exp\left(i \, p \, cos(\theta) \kappa_i \Delta s\right)\end{aligned}$$

Finally, I integrate over $d\theta$ and simplify.

$$\begin{aligned}\mathbb{W}(\mathbb{P}) &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{\infty} dp \, \frac{p^2}{(2\pi)^2} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(p\right)\right) \frac{2sin(p\kappa_n \Delta s)}{p\kappa_n \Delta s} \\ &= \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^{\infty} dp \, \frac{p}{2\pi^2} \exp\left(\sigma_b \, \Delta s \, \widetilde{\boldsymbol{Z}}\left(p\right)\right) \frac{sin(p\kappa_n \Delta s)}{\kappa_n \Delta s}\end{aligned} \tag{A.16}$$

I have thus obtained the final, exact form of the path weight function $W(\mathbb{P})$.

A.4 Regularized Path Weight Function

Unfortunately, this form of the path weight function is numerically unstable, as when I take $p \to \infty$, I expect that $\widetilde{\boldsymbol{Z}}(p) \to 0$. However, as I exponentiate $\widetilde{\boldsymbol{Z}}(p)$, the entire form does not go to 0. Thus, the previous work introduced a regularized form of the path weighting formula, introducing a small, Gaussian blurring applied to the unscattered perturbation.

$$\mathbb{W}^R\left(\mathbb{P}\right) = \prod_{i=0}^{M-1} e^{-\sigma_t \Delta s} \int_0^\infty dp\, \frac{p}{2\pi^2} \exp\left(\sigma_b\, \Delta s\, \left(\widetilde{\boldsymbol{Z}}\left(p\right) - \frac{\epsilon}{2}|p^2|\right)\right) \frac{sin(p\kappa_n \Delta s)}{\kappa_n \Delta s} \tag{A.17}$$

As this form is numerically stable for $\epsilon \neq 0$, this is instead used when numerically solving. Additionally, as $\epsilon \to 0$, the numerical solution converges to the exact FPI-RT solution.

APPENDIX B

DERIVATION OF PATH SAMPLING PDF

I want to normalize path sampling within my numerical solver accurately concerning the delta function. In general, the portion of the solver I provide a normalization for is:

$$G = \int [D\mathbb{P}]\, \delta^3 \left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S \hat{\boldsymbol{\beta}}(s') ds \right)\, \delta^3 \left(\hat{\boldsymbol{\beta}}(0) - \hat{\boldsymbol{\omega}}_S \right)\, \delta^3 (\hat{\boldsymbol{\beta}}(s) - \hat{\boldsymbol{\omega}}_R) \tag{B.1}$$
$$\times\, \delta \left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S \right)\, \delta \left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R \right)\, W(\mathbb{P})$$

Which can be written in the discrete path form as:

$$G_M = \int [\mathbb{P}] \delta^3 \left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s \sum_{i=0}^{M-1} \hat{\boldsymbol{\beta}}_i \right) \delta^3 \left(\hat{\boldsymbol{\beta}}_0 - \hat{\boldsymbol{\omega}}_S \right) \delta^3 \left(\hat{\boldsymbol{\beta}}_{M-1} - \hat{\boldsymbol{\omega}}_R \right) \tag{B.2}$$
$$\times\, \delta \left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S \right)\, \delta \left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R \right)\, W(\mathbb{P})$$

I take $\mathbb{P}$ to represent a full path and $\int [D\mathbb{P}]$ to represent an integration over valid path space. In addition, take note of the five delta functions operating on the paths. These act as boundary constraints for valid paths, and all paths contributing to the final path integration must enforce these boundary conditions. I must take care to account for them in my derivation of a path sampling PDF.

This FPI can be estimated by:

$$G = \frac{1}{N}\sum_{i=0}^{N-1} \delta^3\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s\hat{\boldsymbol{\omega}}_R - \Delta s\hat{\boldsymbol{\omega}}_S - \Delta s\sum_{i=1}^{M-1}\hat{\boldsymbol{\beta}}_i\right)\ \mathbb{W}(\mathbb{P}_i) \tag{B.3}$$

However, the delta function must be normalized to be a valid Monte Carlo estimator. Thus, I must find a normalizing term K_M for a Monte Carlo estimator of paths with M segments.

$$G = K_M\frac{1}{N}\sum_{i=0}^{N-1} \delta^3\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s\hat{\boldsymbol{\omega}}_R - \Delta s\vec{\boldsymbol{x}}_S - \Delta s\sum_{i=1}^{M-1}\hat{\boldsymbol{\beta}}_i\right)\ \mathbb{W}(\mathbb{P}_i) \tag{B.4}$$

To simplify, I can combine all delta functions into a single delta function. I also ignore the path weighting function, as it does not contribute to the sampling probability of a single path.

$$F_M(\hat{\boldsymbol{\beta}}) = \int\prod_{i=1}^{M-3} \delta^3\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s\hat{\boldsymbol{\omega}}_R - \Delta s\hat{\boldsymbol{\omega}}_S - \Delta s\sum_{i=1}^{M-3}\hat{\boldsymbol{\beta}}_i - \Delta s\hat{\boldsymbol{\beta}}\right) \tag{B.5}$$

where I can take

$$\begin{gathered}\vec{\boldsymbol{q}} = \frac{\vec{\boldsymbol{x}}_S - \vec{\boldsymbol{x}}_R}{\Delta s} - \hat{\boldsymbol{\omega}}_S - \hat{\boldsymbol{\omega}}_R \\ F_M(\hat{\boldsymbol{\beta}}) = \frac{1}{\Delta s^3}\int\prod_{i=1}^{M-3}\delta^3\left(\vec{\boldsymbol{q}} - \sum_{i=1}^{M-3}\hat{\boldsymbol{\beta}}_i - \hat{\boldsymbol{\beta}}\right)\end{gathered} \tag{B.6}$$

Now, I can find the normalizing term K_M of an FPI of paths with M segments to be

$$K_M = \int d\Omega F_M(\hat{\boldsymbol{\beta}}) \tag{B.7}$$

B.1 Notable Properties of the Delta Function

I use a few notable delta function properties in my following derivation, which I list below.

$$\begin{aligned}
\delta^{(3)}(\vec{\boldsymbol{x}}\,a) &= \frac{1}{|a|^3}\delta^{(3)}(\vec{\boldsymbol{x}}) \\
\delta^{(3)}(\vec{\boldsymbol{x}}-\vec{\boldsymbol{y}}) &= \frac{\delta^{(1)}\left(|\vec{\boldsymbol{x}}|-|\vec{\boldsymbol{y}}|\right)}{|\vec{\boldsymbol{x}}|^2}\delta^{(2)}(\hat{\vec{\boldsymbol{x}}}-\hat{\vec{\boldsymbol{y}}}) \\
\delta(f(\vec{\boldsymbol{x}})) &= \frac{\delta(\vec{\boldsymbol{x}}-\vec{\boldsymbol{x}}_0)}{|\frac{df}{dx}(\vec{\boldsymbol{x}}_0)|}
\end{aligned}$$

B.2 Small Path Segment Derivations

I now focus on the case where I have three total segments. I begin with three segments as the cases of one and two-segment paths are uninteresting.

$$\begin{aligned}
F_3(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3}\delta^3(\vec{\boldsymbol{q}}-\hat{\boldsymbol{\beta}}) \\
K_3 &= \int d\Omega F_3(\hat{\boldsymbol{\beta}}) \\
K_3 &= \frac{1}{(\Delta s)^3}\int d\Omega\delta^3(\vec{\boldsymbol{q}}-\hat{\boldsymbol{\beta}}) \\
K_3 &= \frac{1}{(\Delta s)^3}\int d\Omega\frac{\delta^{(1)}(|\vec{\boldsymbol{q}}|-|\hat{\boldsymbol{\beta}}|)}{|\vec{\boldsymbol{q}}|^2}\delta^{(2)}(\hat{\vec{\boldsymbol{q}}}-\hat{\boldsymbol{\beta}}) \\
K_3 &= \frac{1}{(\Delta s)^3}\delta^{(1)}(|\vec{\boldsymbol{q}}|-1)\int d\Omega\delta^{(2)}(\hat{\vec{\boldsymbol{q}}}-\hat{\boldsymbol{\beta}}) \\
K_3 &= \frac{1}{(\Delta s)^3}\delta^{(1)}(|\vec{\boldsymbol{q}}|-1)
\end{aligned} \tag{B.8}$$

Note, I can reduce $\int d\Omega\delta^{(2)}(\hat{\vec{\boldsymbol{q}}}-\hat{\boldsymbol{\beta}})$ to one, as the integration over $d\Omega$ leads to a single value for $\hat{\boldsymbol{\beta}}$ which equals $\vec{\boldsymbol{q}}$. Additionally, I have that $\vec{\boldsymbol{q}}$ must be of length one due the the constraint $\delta^{(1)}(|\vec{\boldsymbol{q}}|-1)$.

In this case, the normalizing term reduces to a one-dimensional delta function that encodes the boundary conditions of the FPI. With only three segments, the first and last segments are locked and enforced by the boundary conditions. This leaves a single free segment that must exactly connect the first and last segment. The delta function in F_3 encodes this single valid orientation of the free segment.

I now continue with the derivation of F_4.

$$\begin{aligned} F_4(\hat{\boldsymbol{\beta}}) &= \int d\Omega' \; F_3(\hat{\boldsymbol{\beta}}' + \hat{\boldsymbol{\beta}}) \\ F_4(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\Omega' \; \delta^3(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}') \end{aligned} \tag{B.9}$$

As $\hat{\boldsymbol{\beta}}'$ is a unit vector, $|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}| = 1$.

$$\begin{aligned} F_4(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \, \delta^{(1)}(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}| - 1) \int d\Omega' \; \delta^{(2)} \left(\frac{\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}}{|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}|} - \hat{\boldsymbol{\beta}}' \right) \\ F_4(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \, \delta^{(1)}(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}| - 1) \end{aligned} \tag{B.10}$$

This reduction is due to the definition of the integration of a delta function.

$$\begin{aligned} K_4 &= \int d\Omega F_4(\hat{\boldsymbol{\beta}}) \\ K_4 &= \frac{1}{(\Delta s)^3} \int d\Omega \; \delta^{(1)}(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}| - 1) \end{aligned} \tag{B.11}$$

I can further reduce this by:

$$\begin{gathered} |\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}| = \sqrt{|\vec{\boldsymbol{q}}|^2 + 1 - 2\left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\beta}}\right)} \\ d\Omega = d(cos\theta)d\phi \\ K_4 = \frac{1}{(\Delta s)^3} \int d\phi d(cos\theta) \delta^{(1)}(\sqrt{|\vec{\boldsymbol{q}}|^2 + 1 - 2|\vec{\boldsymbol{q}}|cos\theta} - 1) \\ x = cos(\theta) \\ K_4 = \frac{2\pi}{(\Delta s)^3} \int_{-1}^{1} dx \; \delta^{(1)}(\sqrt{|\vec{\boldsymbol{q}}|^2 + 1 - 2|\vec{\boldsymbol{q}}|x} - 1) \end{gathered} \tag{B.12}$$

Then, by the third delta property:

$$
\begin{aligned}
f(x) &= \sqrt{|\vec{\boldsymbol{q}}|^2 + 1 - 2|\vec{\boldsymbol{q}}|x} - 1 \\
\vec{\boldsymbol{x}}_0 &= |\vec{\boldsymbol{q}}|/2 \\
f'(\vec{\boldsymbol{x}}_0) &= |\vec{\boldsymbol{q}}| \\
K_4 &= \frac{2\pi}{(\Delta s)^3} \int_{-1}^{1} dx \, \frac{\delta^{(1)}(x - |\vec{\boldsymbol{q}}|/2)}{|\vec{\boldsymbol{q}}|} \\
K_4 &= \frac{2\pi}{(\Delta s)^3} \frac{\mathcal{H}(2 - |\vec{\boldsymbol{q}}|)}{|\vec{\boldsymbol{q}}|}
\end{aligned}
\tag{B.13}
$$

In this, $\mathcal{H}()$ is the Heaviside step function. I will define a recursive approach for evaluating F_5 now.

$$
\begin{aligned}
F_5(\hat{\boldsymbol{\beta}}) &= \int d\Omega' \, F_4(\hat{\boldsymbol{\beta}}' + \hat{\boldsymbol{\beta}}) \\
K_5 &= \int d\Omega \, F_5(\hat{\boldsymbol{\beta}})
\end{aligned}
\tag{B.14}
$$

I continue the derivation below as I cannot numerically integrate $F_4(\hat{\boldsymbol{\beta}}' + \hat{\boldsymbol{\beta}})$ in its current form due to the delta function in F_4.

$$
\begin{aligned}
F_5(\hat{\boldsymbol{\beta}}) &= \int d\Omega' \, F_4(\hat{\boldsymbol{\beta}}' + \hat{\boldsymbol{\beta}}) \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\Omega' \int d\Omega'' \, \delta^3(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}'')
\end{aligned}
\tag{B.15}
$$

As $\hat{\boldsymbol{\beta}}''$ is a unit vector, I must have that $|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}'| = 1$.

$$
\begin{aligned}
F_5(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\Omega' \left[\delta^{(1)}\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}'| - 1\right) \int d\Omega''\, \delta^{(2)}\left(\frac{\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}'}{|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}'|} - \hat{\boldsymbol{\beta}}''\right)\right] \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\Omega' \left[\delta^{(1)}\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}'| - 1\right)\right] \\
\vec{\boldsymbol{r}} &= \vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\Omega' \left[\delta^{(1)}\left(\sqrt{\vec{\boldsymbol{r}}^2 + 1 - 2\vec{\boldsymbol{r}} \cdot \hat{\boldsymbol{\beta}}'} - 1\right)\right] \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{1}{(\Delta s)^3} \int d\phi \int d\cos\theta \left[\delta^{(1)}\left(\sqrt{\vec{\boldsymbol{r}}^2 + 1 - 2|\vec{\boldsymbol{r}}|\cos\theta} - 1\right)\right] \\
x &= cos(\theta) \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{2\pi}{(\Delta s)^3} \int_{-1}^{1} dx \left[\delta^{(1)}\left(\sqrt{\vec{\boldsymbol{r}}^2 + 1 - 2|\vec{\boldsymbol{r}}|x} - 1\right)\right] \\
F_5(\hat{\boldsymbol{\beta}}) &= \frac{2\pi}{(\Delta s)^3} \frac{\mathcal{H}(2 - |\vec{\boldsymbol{r}}|)}{|\vec{\boldsymbol{r}}|}
\end{aligned} \tag{B.16}
$$

As I can now numerically integrate F_5, I can additionally integrate K_5

$$
K_5 = \int d\Omega F_5(\hat{\boldsymbol{\beta}}) \tag{B.17}
$$

$$
K_5 = \int d\Omega \, \frac{2\pi}{(\Delta s)^3} \frac{\mathcal{H}(2 - |\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}|)}{|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}|} \tag{B.18}
$$

APPENDIX C

VALIDATION OF PATH-PERTURBATION SOLVER[*]

In my Annals of Nuclear Energy publication, I validated my novel numerical solver by evaluating my method against two tests that parallel those in the previous work [14], as well as one novel test that measures how well my solver captures the expected distribution of tangents in all generated paths [36]. As all three tests focus on the behavior of the numerical solver for a single transport kernel $G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right)$, rather than an integration over transport kernel evaluations for different boundary conditions, all paths in a single numerical solver will have one specified arclength S. In this case, absorption acts as an exponential falloff related to the arclength and can thus be ignored. Thus, for all three tests in this section, I focus on scattering only by setting the absorption constant of the scene to 0.0.

C.1 Statistical Stationarity of the Geometric Perturbation

The first method used for verification analyzes the statistical stationarity of path weights over long, sequential sequences of path weights, referred to as weight slices, randomly selected from within a much larger sequence. This test measures the ability of the proposed solver to generate a sequence of paths that converges to the same FPI, regardless of the history of path perturbation. To perform the test, a large number of paths are generated and weighted, producing the path weights $W = \{\mathbb{W}(\mathbb{P}_0), ..., \mathbb{W}(\mathbb{P}_{N-1}\}$. Then, K random starting indices are selected within the generated path weights, each representing the starting location for a slice of path weights of length L from I. For each weight slice S_i, the weight of that slice is computed by summing the L path weights starting at the weight slice's corresponding starting index, K_i. These generated values are then presented in a histogram. A set of 10^8 path weights were generated, and weight slices of lengths

[*]Reprinted with permission from "Path integral radiative transfer via polyline representation allowing GPU implementation" by Brennen Taylor, John Keyser, Jerry Tessendorf, 2022. Annals of Nuclear Energy, 173, 109098, Copyright 2022 by Name of Elsevier.

1000, 2000 and 5000 are presented in Figures C.1, C.2 and C.3. Note that the distribution of weight slices in the histograms is distinctly log-normal in form.

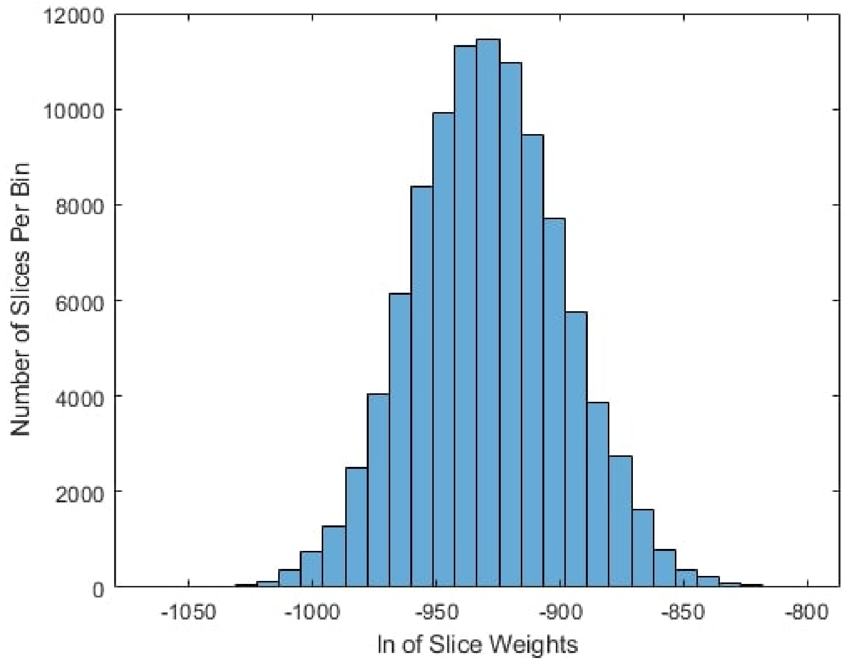


Figure C.1: Histogram of 100,000 random slices of 1000 consecutive path weights. X-axis represents the natural log of slice weights.

I found that as the weight slice length L increased from 1000 to 5000, the width μ of the log-normal fits decreased, corresponding to a convergence of the calculated weight slice weights to the same solution; this behavior is consistent with the previous solver[14]. From this, I can conclude that no matter the initial starting path, my method can generate sequences of paths that converge to the same solution.

C.2 Convergence of the Path Integral

My second validation method focuses on the convergence of my path integral calculation as the number of paths included increases. While the prior test focused on measuring my path generation algorithm's ability to generate sequences of paths whose path integral converges to the solution,

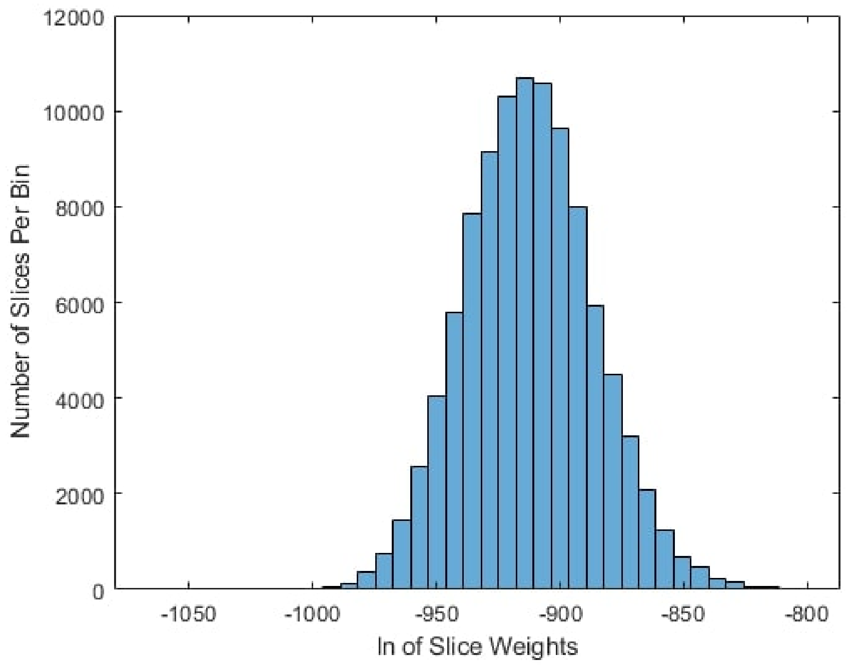


Figure C.2: Histogram of 100,000 random slices of 2000 consecutive path weights. X-axis represents the natural log of slice weights.

this test focuses on the entire numerical solver's ability to converge to a solution. For this experiment, many paths, $\mathbb{P}_0, ..., \mathbb{P}_{N-1}$, are generated while simultaneously tracking the running average of all path weights. Figure C.4 shows the result of running the path integral up to a path count of $5 * 10^7$. The path weight average converges on a stable value somewhere between $1 * 10^7$ and $2 * 10^7$ paths generated, consistent with the previous work[14]. It's important to note that convergence remains independent of the absorption parameter, as absorption is modeled as an exponential falloff of path length; thus, the path integral would converge in the same number of paths, and my absorption setting of 0.0 has no impact on this conclusion.

C.3 Uniform Sampling of Paths

My third and novel validation method measures my numerical solution's ability to sample path space without bias. This analysis involves breaking the tangent $\hat{\boldsymbol{T}}_i$ of each segment of each path in a numerical solve run into two components, $e_{0,i}$ and θ_i. $e_{0,i}$ is the component of tangent $\hat{\boldsymbol{T}}_i$ along

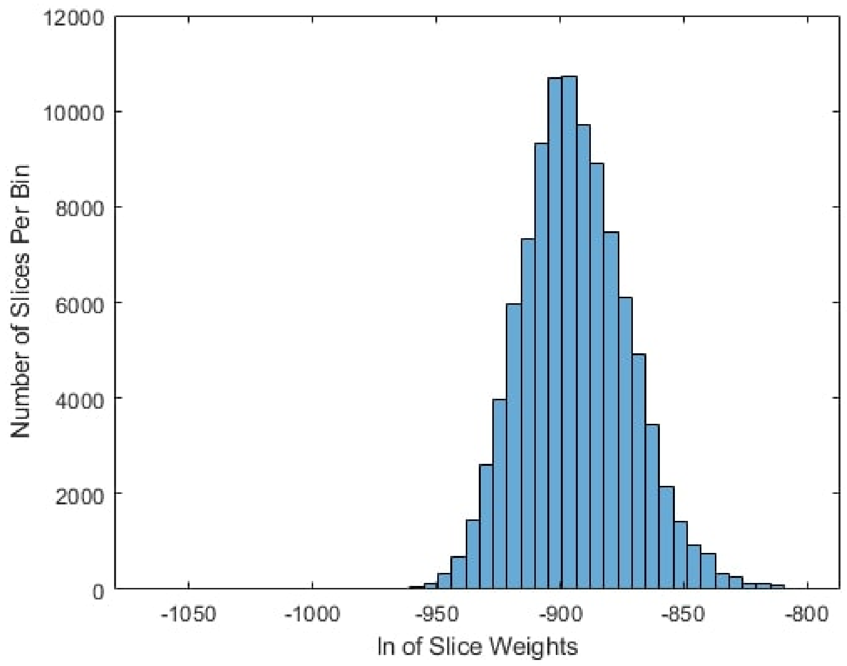


Figure C.3: Histogram of 100,000 random slices of 5000 consecutive path weights. X-axis represents the natural log of slice weights.

a principal axis $\vec{z} = (\vec{x}_R - \vec{x}_S)/\Delta s - \hat{\omega}_R - \hat{\omega}_S$ and is defined in the range $[-1, 1]$. θ_i is the angle around the principal axis $\vec{z}$ and is defined in the range $[0, 2\pi]$. It's important to note that the path sample space is environment specific, defined by the boundary conditions $\mathbb{R}$. For validation, I carefully constructed an environment such that the boundary conditions are along a single axis; in this case, $\mathbb{R} = \{\vec{x}_S = \{0,0,0\}, \hat{\omega}_S = \{1,0,0\}, \vec{x}_R = \{10,0,0\}, \hat{\omega}_R = \{1,0,0\}\}$. This allows for the analytical derivation of the ground truth expected distribution of sampled tangents. Additionally, due to the symmetry of the environment, the theta distribution is known to be uniform and thus is not analyzed further. Instead, I focus on the distribution of e_0 values.

As seen in Figure C.5, my method closely captures the expected distribution of e_0 values for all target arclength values.

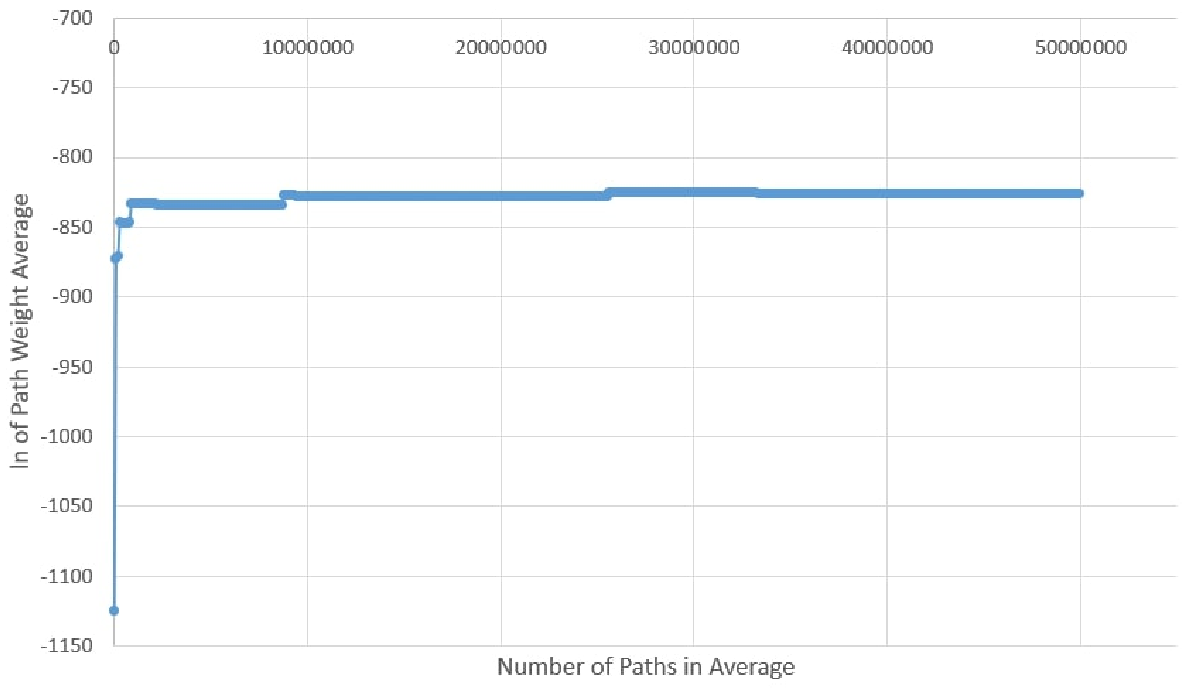
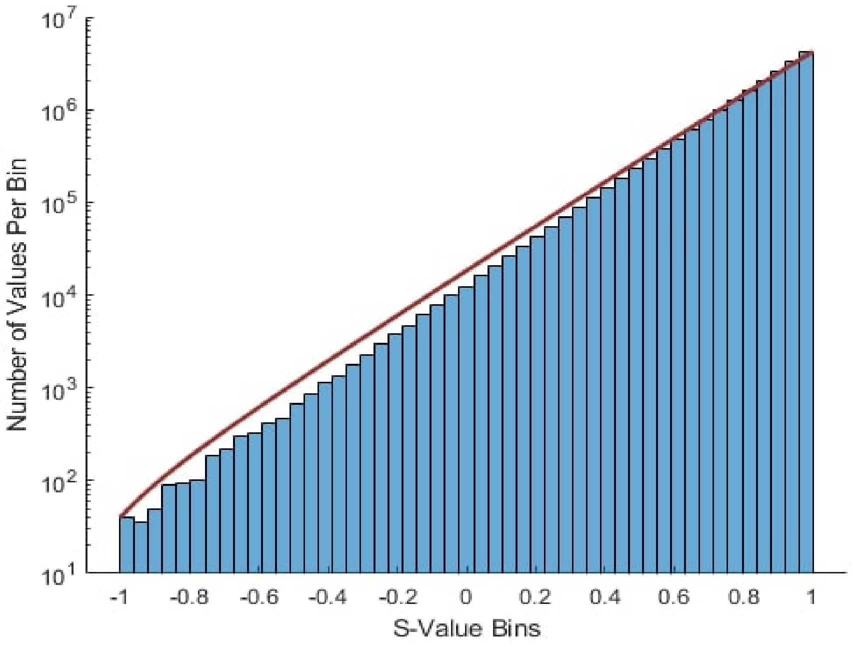


Figure C.4: Convergence of the average path weight versus the number of paths in the average. Y-axis represents the natural log of average weight.

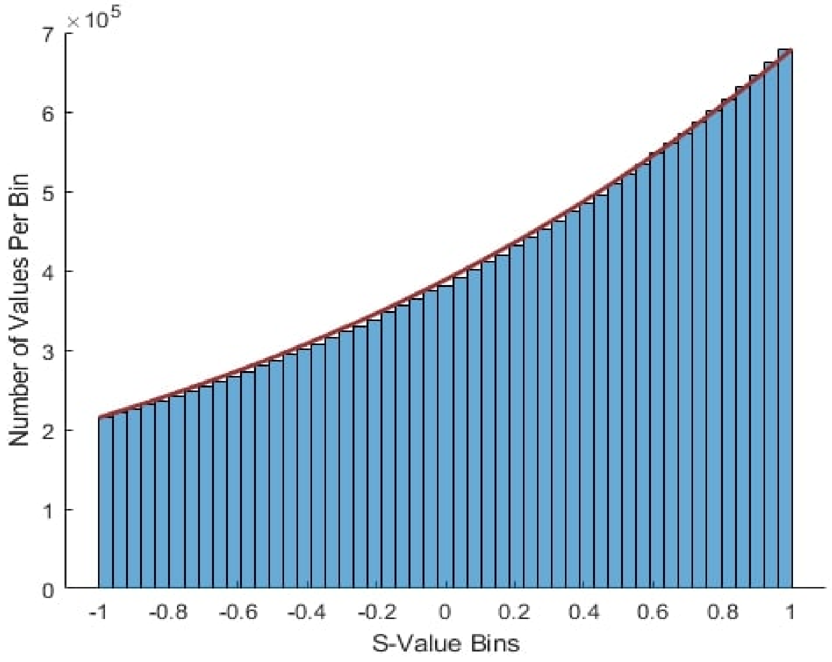
C.4 Functional Forms

I provide the following functional form I utilized for calculating the ground truth uniform sampling of paths. This ground truth represents the expected distribution of tangents for all curves of specified arclength generated for the carefully constructed environment in which the start and end directions all lie on the $\hat{\vec{x}}$-axis and the start and end positions are located 10 units apart.

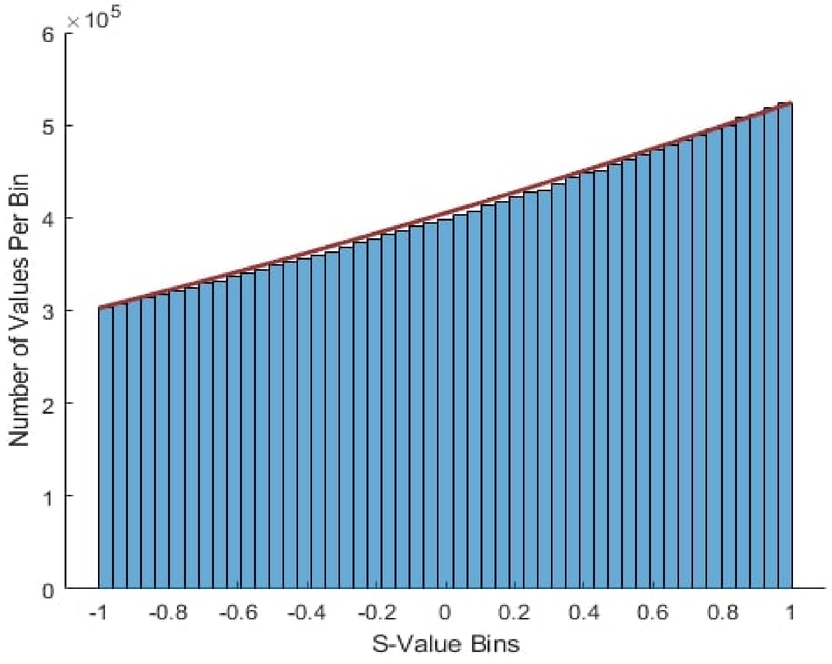
I desire to develop a PDF for selecting a single tangent weight $\hat{\boldsymbol{T}}$ for any path $\mathbb{P}$ with boundary conditions $\mathbb{R}$. For this, I focus only on the delta function portion of the transport kernel and can ignore the weighting portion, as it simply acts on the path and does not restrict the path.



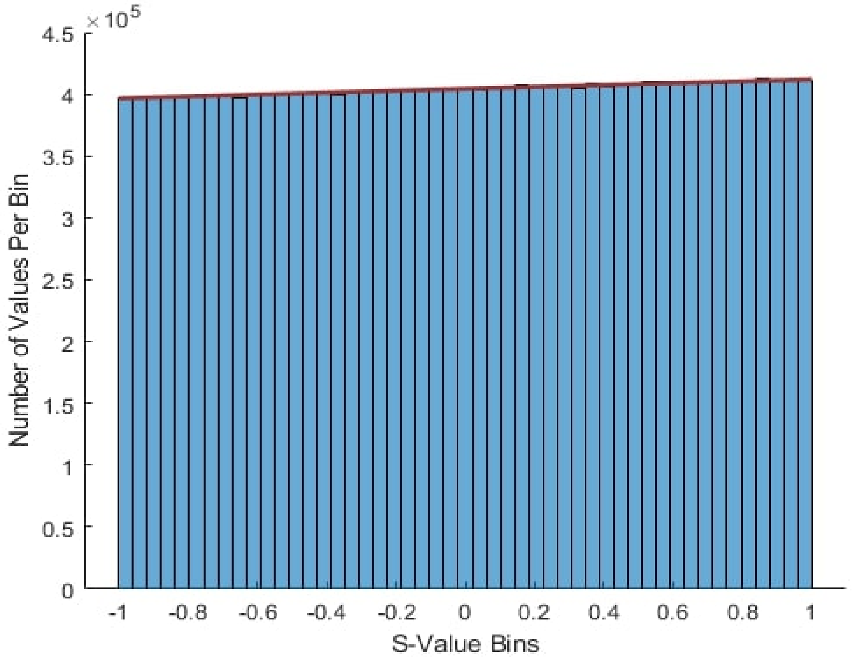
Arclength of 12. Note the Y-axis log scale.



Arclength of 50. Note the Y-axis log scale.



Arclength of 100. Note the Y-axis log scale.



Arclength of 1000. Note the Y-axis log scale.

Figure C.5: Visualization of ground truth (red) and histogram of the experimental (blue) distribution of e_0 values for experiment runs of arclength $S=$ 12, 50, 100, and 1000 units. The fit improves as the arclength increases from the minimum possible arclength defined by the experiment constraints toward infinity.

$$
\begin{aligned}
G\left(S,\vec{\boldsymbol{x}}_R,\hat{\boldsymbol{\omega}}_R,\vec{\boldsymbol{x}}_S,\hat{\boldsymbol{\omega}}_S\right)' &= \delta\left(\vec{\boldsymbol{x}}_R-\vec{\boldsymbol{x}}_S-\int\hat{\boldsymbol{\beta}}\right)\delta\left(\hat{\boldsymbol{\beta}}(0)-\hat{\boldsymbol{\omega}}_S\right)\delta\left(\hat{\boldsymbol{\beta}}(S)-\hat{\boldsymbol{\omega}}_R\right)\\
&\times\delta\left(\vec{\boldsymbol{r}}(0)-\vec{\boldsymbol{x}}_S\right)\ \delta\left(\vec{\boldsymbol{r}}(S)-\vec{\boldsymbol{x}}_R\right)\\
&=\left(\frac{1}{\Delta s}\right)^3\int\prod_{i=1}^{M-2}\delta\left(\vec{\boldsymbol{q}}-\prod_{i=1}^{M-2}\hat{\boldsymbol{\beta}}_i\right)
\end{aligned}
$$

I can then take the PDF of selecting a single valid tangent $\hat{\boldsymbol{\beta}}$ to be

$$
PDF_M(\hat{\boldsymbol{\beta}})=\left(\frac{1}{\Delta s}\right)^3\int\prod_{i=2}^{M-2}\delta\left(\vec{\boldsymbol{q}}-\hat{\boldsymbol{\beta}}-\prod_{i=2}^{M-2}\hat{\boldsymbol{\beta}}_i\right)
$$

I refer to the Heaviside step function as $H(x)$. I define a function f_M of a given order M, as well as two base cases f_3 and f_4. In this function, each increase in order represents an increase in the number of discrete segments per path by 1.

$$
\begin{aligned}
f_M(r) &= \int_{|r-1|}^{r+1} dr' f_{M-1}(r')\\
f_3(r) &= H(3-r)(3-r)-3H(1-r)(1-r)\\
f_4(r) &= H(4-r)\frac{(2-r)^2}{2}-H(2-r)\frac{(2-r)^2}{2}+H(2-r)\frac{3r^2}{2}
\end{aligned}
\tag{C.1}
$$

I then define a function $L_M(e_0,\vec{\boldsymbol{q}})$, which for a specified arclength S, and alignment parameter e_0 between -1 and 1, calculates the relative likelihood that a tangent $\hat{\boldsymbol{T}}$ is sampled such that $\hat{\boldsymbol{T}}\cdot\vec{\boldsymbol{q}}=e_0$. Note the arclength parameter S and the alignment parameter e_0 are unrelated.

$$\begin{aligned} \vec{\boldsymbol{q}} &= \frac{\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S}{\Delta s} - \hat{\boldsymbol{\omega}}_R - \hat{\boldsymbol{\omega}}_S \\ L_M\left(e_0, |\vec{\boldsymbol{q}}|\right) &= \frac{(2\pi)^{M-1} f_M\left(\sqrt{2e_0 q + q^2 + 1}\right)}{\sqrt{2e_0 q + q^2 + 1}} \end{aligned} \tag{C.2}$$

Finally, I evaluate $L_{M=200}$ for specified arclength values of 12, 50, 100, and 1000, and on a full range of $e_0 = [-1, 1]$, generating the expected values for my uniform sampling of paths analysis in section 4.3 which accounts for all possible segment tangent directions of the specified arclength. I then scale this ground truth to fit the minimum and maximum of my experimental data as I lack a true normalizing term. This validation method demonstrates that the presented perturbation method generates sets of paths with the expected distribution of tangents.

APPENDIX D

DERIVATION OF SIMPLIFIED WEIGHT ANALYTICAL SOLUTION

D.1 General Construction

In this chapter, I discuss deriving an exact, analytic solution of the path integral for two, three, and four-segment paths weighted by a simplified weighting function. Additionally, I present the recursive manner in which the solution for higher segment count paths could be computed recursively using the solutions for smaller segment count paths.

I begin with a modified path integral form with a simplified weighting kernel.

$$\begin{aligned} G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = \int d\Omega\, \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S ds' \hat{\boldsymbol{\beta}}(s')\right) \\ \delta\left(\hat{\boldsymbol{\beta}}(0) - \hat{\boldsymbol{\omega}}_S\right)\, \delta\left(\hat{\boldsymbol{\beta}}(S) - \hat{\boldsymbol{\omega}}_R\right) \\ \times\, \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right)\, \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R\right) \\ \times\, exp\left(-\frac{1}{\Delta s \mu \sigma_b} \int_0^S ds' |\hat{\boldsymbol{\beta}}(s')|^2\right) \end{aligned} \tag{D.1}$$

I can easily simplify this by defining $\alpha = 1/(\Delta s \mu \sigma_b)$, leaving

$$G\left(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S\right) = \int d\Omega\, \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \int_0^S ds' \hat{\boldsymbol{\beta}}(s')\right) \tag{D.2}$$
$$\times\, \delta\left(\hat{\boldsymbol{\beta}}(0) - \hat{\boldsymbol{\omega}}_S\right)\, \delta\left(\hat{\boldsymbol{\beta}}(s) - \hat{\boldsymbol{\omega}}_R\right)$$
$$\times\, \delta\left(\vec{\boldsymbol{r}}(0) - \vec{\boldsymbol{x}}_S\right)\, \delta\left(\vec{\boldsymbol{r}}(S) - \vec{\boldsymbol{x}}_R\right)$$
$$\times\, exp\left(-\alpha \int_0^S ds' |\hat{\boldsymbol{\beta}}(s')|^2\right)$$

I now discretize the formulation of G for M segments, G_M.

$$G_M(S, \vec{\boldsymbol{x}}_R, \hat{\boldsymbol{\omega}}_R, \vec{\boldsymbol{x}}_S, \hat{\boldsymbol{\omega}}_S) = \int \left(\prod_{i=1}^{M-2} d\Omega_i\right) \delta\left(\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S - \Delta s(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R) - \Delta s \sum_{i=1}^{M-2} \hat{\boldsymbol{\beta}}_i\right) \tag{D.3}$$
$$\times\, exp\left(-\alpha \sum_{i=0}^{M-2} \left(1 - \hat{\boldsymbol{\beta}}_{i+1} \cdot \hat{\boldsymbol{\beta}}_i\right)\right)$$

Note I drop all other boundary condition-enforcing delta functions, as these only enforce the placement of the FPI-RT in space and have no impact on the numerical solution. Defining the following

$$\vec{\boldsymbol{q}} = \left(\frac{\vec{\boldsymbol{x}}_R - \vec{\boldsymbol{x}}_S}{\Delta s} - \hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\omega}}_R\right) \tag{D.4}$$

I simplify G_M further by substituting $\vec{\boldsymbol{q}}$ in, as well as by distributing $-\alpha$

$$G_M\left(\vec{q},\hat{\boldsymbol{\beta}}\right)=\frac{exp(-\alpha(M-1))}{\Delta s^3}\int\left(\prod_{i=1}^{M-2}d\Omega_i\right)\delta\left(\vec{\boldsymbol{q}}-\sum_{i=1}^{M-2}\hat{\beta}_i\right) \times exp\left(\alpha\sum_{i=0}^{M-2}\left(\hat{\boldsymbol{\beta}}_{i+1}\cdot\hat{\boldsymbol{\beta}}_i\right)\right) \tag{D.5}$$

D.2 Analytic Derivation of Small Segment Count Paths

D.2.1 Two Segments

First, I have the equation for G_2

$$G_2\left(\vec{\boldsymbol{q}}\right)=\frac{\delta\left(\vec{\boldsymbol{q}}\right)}{(\Delta s)^3}\,exp\left(-\alpha\right)\;exp\left(\alpha\left(\hat{\boldsymbol{\beta}}_1\cdot\hat{\boldsymbol{\beta}}_0\right)\right) \tag{D.6}$$

As this represents the analytical solution for paths with exactly two segments, I must have that in a valid environment, $\hat{\boldsymbol{\beta}}=\hat{\boldsymbol{\omega}}_S$ and $\hat{\boldsymbol{\beta}}_1=\hat{\boldsymbol{\omega}}_R$. In doing so, I obtain the following.

$$G_2\left(\vec{\boldsymbol{q}}\right)=\frac{\delta\left(\vec{\boldsymbol{q}}\right)}{(\Delta s)^3}\,exp\left(-\alpha\right)\;exp\left(\alpha\left(\hat{\boldsymbol{\omega}}_S\cdot\hat{\boldsymbol{\omega}}_R\right)\right) \tag{D.7}$$

However, to derive the solution for higher path counts, I allow segment $\hat{\boldsymbol{\beta}}_1$ to be specified freely.

$$G_2\left(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}}\right)=\frac{\delta\left(\vec{\boldsymbol{q}}\right)}{(\Delta s)^3}\,exp\left(-\alpha\right)\;exp\left(\alpha\left(\hat{\boldsymbol{\beta}}\cdot\hat{\boldsymbol{\omega}}_S\right)\right) \tag{D.8}$$

I can exactly reproduce the non-recursive case by calling $G_2\left(\vec{\boldsymbol{q}},\hat{\boldsymbol{\omega}}_R\right)$.

D.2.2 Three Segments

I can then use the recursive relation to derive G_3 using G_2.

I have

$$G_3\left(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}\right) = e^{-\alpha} \int d\Omega' \, G_2\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}', \hat{\boldsymbol{\beta}}'\right) \, exp\left(\alpha\left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}'\right)\right) \tag{D.9}$$

I plug in G_2 to obtain

$$\begin{aligned} G_3\left(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}\right) &= e^{-\alpha} \int d\Omega' \, \frac{1}{(\triangle s)^3} \, \delta\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'\right) \, exp\left(-\alpha\right) \qquad \text{(D.10)} \\ &\times exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \hat{\boldsymbol{\omega}}_S\right)\right) \, exp\left(\alpha\left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}'\right)\right) \\ &= \frac{e^{-2\alpha}}{(\Delta s)^3} \int d\Omega' \, \delta\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'\right) \, exp\left(\alpha \, \hat{\boldsymbol{\beta}}' \cdot \left(\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\omega}}_S\right)\right) \qquad \text{(D.11)} \end{aligned}$$

Next, I use the following definition of the three-dimensional delta function.

$$\delta\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}\right) = \delta\left(|\vec{\boldsymbol{q}}| - |\hat{\boldsymbol{\beta}}|\right) \delta^{(2)}\left(\hat{q} - \hat{\boldsymbol{\beta}}\right) \tag{D.12}$$

I use this definition for simplification and $q = |\vec{\boldsymbol{q}}|$, yielding the following.

$$G_3\left(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}}\right)=\frac{e^{-2\alpha}}{(\Delta s)^3}\int d\Omega'\,\delta\left(|\vec{\boldsymbol{q}}|-|\hat{\boldsymbol{\beta}}'|\right)\delta^{(2)}\left(\hat{q}-\hat{\boldsymbol{\beta}}'\right)\;exp\left(\alpha\,\hat{\boldsymbol{\beta}}'\cdot\left(\hat{\boldsymbol{\beta}}+\hat{\boldsymbol{\omega}}_S\right)\right) \tag{D.13}$$

$$=\frac{e^{-2\alpha}}{(\Delta s)^3}\int d\Omega'\,\delta\left(q-1\right)\delta^{(2)}\left(\hat{q}-\hat{\boldsymbol{\beta}}'\right)\;exp\left(\alpha\,\hat{\boldsymbol{\beta}}'\cdot\left(\hat{\boldsymbol{\beta}}+\hat{\boldsymbol{\omega}}_S\right)\right) \tag{D.14}$$

Referencing the three-segment path environment 5.2c, it is easy to see that again, the environment is constrained such that only one valid path exists. One can see in the above equation that the second delta function enforces that $\hat{q}$ and $\hat{\boldsymbol{\beta}}'$ must be equal. Thus, when integrating over $d\Omega$, I have that $\hat{q}$ must equal $\hat{\boldsymbol{\beta}}'$. Using this, I continue to simplify

$$G_3(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}})=\frac{e^{-2\alpha}}{(\Delta s)^3}\,\delta\left(q-1\right)\;exp\left(\alpha\left(\hat{q}\cdot\left(\hat{\boldsymbol{\omega}}_S+\hat{\boldsymbol{\beta}}\right)\right)\right) \tag{D.15}$$

Thus, I have arrived at my final form for G_3. As expected, the final form is constrained to only allow for a single valid path, seen with the presence of the delta function.

D.2.3 Four Segments

I continue and derive the form for G_4. As seen in 5.3, this is the smallest segment count path that allows for more than one valid path. If solving numerically, I would need to integrate over all possible paths. However, it is also possible to derive the exact analytical solution.

I begin with the following recursive relation.

$$G_4(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}})=e^{-\alpha}\int d\Omega' G_3\left(\vec{\boldsymbol{q}}-\hat{\boldsymbol{\beta}}',\hat{\boldsymbol{\beta}}'\right)\;exp\left(\alpha\left(\hat{\boldsymbol{\beta}}\cdot\hat{\boldsymbol{\beta}}'\right)\right) \tag{D.16}$$

Substituting the equation for G_3 and performing basic simplification, I obtain

$$\begin{aligned} G_4(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) &= e^{-\alpha} \int d\Omega' \, \frac{e^{-2\alpha}}{(\Delta s)^3} \delta \left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1 \right) \\ &\times exp \left(\alpha \left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} \right) \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}' \right) \right) \; exp \left(\alpha \left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}' \right) \right) \\ &= \frac{e^{-3\alpha}}{(\Delta s)^3} \int d\Omega' \, \delta \left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1 \right) \\ &\times exp \left(\alpha \left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}} \right) \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}' \right) \right) \; exp \left(\alpha \left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}' \right) \right) \end{aligned} \tag{D.17}$$

Next, using the following

$$\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}' \right) \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}' \right) = (\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S) + \left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\beta}}' \right) - \left(\hat{\boldsymbol{\beta}}' \cdot \hat{\boldsymbol{\omega}}_S \right) - |\hat{\boldsymbol{\beta}}'|^2 \tag{D.18}$$

$$= (\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S) + \left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\beta}}' \right) - \left(\hat{\boldsymbol{\beta}}' \cdot \hat{\boldsymbol{\omega}}_S \right) - 1 \tag{D.19}$$

I continue to simplify the previous form of G_4.

$$
\begin{aligned}
G_4\left(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}\right) &= \frac{e^{-3\alpha}}{(\Delta s)^3} \int d\Omega' \, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right) \\
&\times exp\left(\alpha\left(\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}\right) \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}'\right)\right) \; exp\left(\alpha\left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}'\right)\right) \\
&= \frac{e^{-3\alpha}}{(\Delta s)^3} \int d\Omega' \, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right) \\
&\times exp\left(\alpha\left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}' + \vec{\boldsymbol{q}} \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}'\right) - \left(\hat{\boldsymbol{\beta}}' \cdot \hat{\boldsymbol{\omega}}_S\right) - 1\right)\right) \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3} \int d\Omega' \, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right) \\
&\times exp\left(\alpha\left(\hat{\boldsymbol{\beta}} \cdot \hat{\boldsymbol{\beta}}' + \vec{\boldsymbol{q}} \cdot \left(\hat{\boldsymbol{\omega}}_S + \hat{\boldsymbol{\beta}}'\right) - \left(\hat{\boldsymbol{\beta}}' \cdot \hat{\boldsymbol{\omega}}_S\right)\right)\right) \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3} \\
&\times exp\left(\alpha\left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S\right)\right) \int d\Omega' \, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right) \; exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \left(\hat{\boldsymbol{\beta}} + \vec{\boldsymbol{q}} - \hat{\boldsymbol{\omega}}_S\right)\right)\right)
\end{aligned} \tag{D.20}
$$

To continue my derivation, next focus on the delta function $\delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right)$.

$$0 = |\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1 \tag{D.21}$$

$$1 = |\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'|^2 \tag{D.22}$$

$$1 = |\vec{\boldsymbol{q}}|^2 + |\hat{\boldsymbol{\beta}}'|^2 - 2\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}} \tag{D.23}$$

$$1 = q^2 + 1 - 2\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}} \tag{D.24}$$

$$0 = q^2 - 2\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}} \tag{D.25}$$

Next, I define $r = \hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}}$, and substitute to obtain

$$0 = q^2 - 2r \tag{D.26}$$

$$2r = q^2 \tag{D.27}$$

$$r = \frac{q^2}{2} \tag{D.28}$$

Thus, $\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}} = \frac{q^2}{2}$. However, it is important to note that this is true only when q is constrained. Focusing on the domain of the dot product $\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}}$, I find that $[-q, q] = \frac{q^2}{2}$. Simplifying, I find that $[-2, 2] = q$. However, as q is the length of the vector q, it must be positive. Thus, $q \leq 2$ must be true. I can enforce this via a Heaviside step function $H(2 - q)$.

I continue simplifying G_4 with these new findings. In particular, I focus on extracting as much out of the solid angle as possible, i.e., any terms that do not depend on $\hat{\boldsymbol{\beta}}'$ or that are enforced by adding $H(2 - q)$.

$$\begin{aligned}
G_4\left(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}\right) &= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp\left(\alpha\left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S\right)\right) & \text{(D.29)}\\
&\times \int d\Omega'\, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right)\, exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \left(\hat{\boldsymbol{\beta}} + \vec{\boldsymbol{q}} - \hat{\boldsymbol{\omega}}_S\right)\right)\right) \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp\left(\alpha\left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S\right)\right) H\left(2 - q\right) & \text{(D.30)}\\
&\times \int d\Omega'\, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right)\, exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}}\right)\right) exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \left(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S\right)\right)\right) \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp\left(\alpha\left(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S\right)\right) H\left(2 - q\right) exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \vec{\boldsymbol{q}}\right)\right) & \text{(D.31)}\\
&\times \int d\Omega'\, \delta\left(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1\right)\, exp\left(\alpha\left(\hat{\boldsymbol{\beta}}' \cdot \left(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S\right)\right)\right) \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp(\alpha(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S)) H(2 - q) exp(\alpha\frac{q^2}{2}) & \text{(D.32)}\\
&\times \int d\Omega'\, \delta(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1)\, exp(\alpha(\hat{\boldsymbol{\beta}}' \cdot (\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S)))
\end{aligned}$$

Next, I continue by focusing on simplifying the solid angle integration. I perform this by splitting and separating the solid angle into two components, $r = \hat{\boldsymbol{\beta}}' \cdot \hat{\vec{\boldsymbol{q}}}$, which represents the amount $\hat{\boldsymbol{\beta}}'$ is aligned with $\vec{\boldsymbol{q}}$, and θ, representing the rotation of $\hat{\boldsymbol{\beta}}'$ around $\vec{\boldsymbol{q}}$. Additionally, let us define a vector perpendicular to $\vec{\boldsymbol{q}}$ called $\vec{\boldsymbol{q}}_{\perp}$, such that $\hat{\boldsymbol{\beta}}' = r\hat{\vec{\boldsymbol{q}}} + \sqrt{1-r^2}\hat{\vec{\boldsymbol{q}}}_{\perp}$.

First, I rewrite G_4 in terms of this separated solid angle.

$$\begin{aligned} G_4(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}) &= \frac{e^{-4\alpha}}{(\Delta s)^3} exp(\alpha(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S))H(2-q)exp(\alpha\frac{q^2}{2}) \qquad \text{(D.33)} \\ &\times \int_0^{2\pi}\int_0^{\pi} sin(\phi)\, d\phi\, d\theta\, \delta(|\vec{\boldsymbol{q}} - \hat{\boldsymbol{\beta}}'| - 1)\, exp(\alpha(\hat{\boldsymbol{\beta}}' \cdot (\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S))) \\ &= \frac{e^{-4\alpha}}{(\Delta s)^3} exp(\alpha(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S))H(2-q)exp(\alpha\frac{q^2}{2}) \qquad \text{(D.34)} \\ &\times \int_0^{2\pi}\int_0^{\pi} sin(\phi)\, d\phi\, d\theta\, \delta(\sqrt{q^2+1-2qr} - 1)\, exp(\alpha(\hat{\boldsymbol{\beta}}' \cdot (\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S))) \end{aligned}$$

I can then rewrite this in terms of r. For this, I also solve for r in $\sqrt{q^2+1-2qr} - 1 = 0$, due to the delta function condition. This results in $r = \frac{q}{2}$, thus the delta function becomes, with the change of variable, $\frac{\delta(r-\frac{q}{2})}{q}$.

$$G_4(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}}) = \frac{e^{-4\alpha}}{(\Delta s)^3}\; exp(\alpha(\vec{\boldsymbol{q}}\cdot\hat{\boldsymbol{\omega}}_S))H(2-q)exp(\alpha\frac{q^2}{2}) \tag{D.35}$$

$$\times\int_0^{2\pi}\int_{-1}^{1} dr\; d\theta\; \frac{\delta(r-\frac{q}{2})}{q}\; exp(\alpha(\hat{\boldsymbol{\beta}}'\cdot(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S)))$$

$$= \frac{e^{-4\alpha}}{(\Delta s)^3}\; exp(\alpha(\vec{\boldsymbol{q}}\cdot\hat{\boldsymbol{\omega}}_S))\frac{H(2-q)}{q}exp(\alpha\frac{q^2}{2}) \tag{D.36}$$

$$\times\int_0^{2\pi} d\theta\; exp(\alpha(\left(\frac{\vec{\boldsymbol{q}}}{2}+\sqrt{1-\frac{q^2}{4}}\hat{q_\perp}\right)\cdot(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S)))$$

$$= \frac{e^{-4\alpha}}{(\Delta s)^3}\; exp(\alpha(\vec{\boldsymbol{q}}\cdot\hat{\boldsymbol{\omega}}_S))\frac{H(2-q)}{q}exp(\alpha\frac{q^2}{2})exp(\alpha(\frac{\vec{\boldsymbol{q}}}{2}\cdot(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S))) \tag{D.37}$$

$$\times\int_0^{2\pi} d\theta\; exp(\alpha(\left(\sqrt{1-\frac{q^2}{4}}\hat{q_\perp}\right)\cdot(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S)))$$

$$= \frac{e^{-4\alpha}}{(\Delta s)^3}\; exp(\alpha(\vec{\boldsymbol{q}}\cdot\hat{\boldsymbol{\omega}}_S))\frac{H(2-q)}{q}exp(\alpha\frac{q^2}{2})exp(\alpha(\frac{\vec{\boldsymbol{q}}}{2}\cdot(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S))) \tag{D.38}$$

$$\times\int_0^{2\pi} d\theta\; exp(\alpha(\left(|\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S|\sqrt{1-\frac{q^2}{4}}\; cos(\theta)\right)))$$

I am now left with the remaining integration of the angle around $\vec{\boldsymbol{q}}$.

$$G_4\left(\vec{\boldsymbol{q}},\hat{\boldsymbol{\beta}}\right) = \frac{e^{-4\alpha}}{(\Delta s)^3}\; exp\left(\alpha\left(\vec{\boldsymbol{q}}\cdot\hat{\boldsymbol{\omega}}_S\right)\right)\frac{H\left(2-q\right)}{q}exp\left(\alpha\frac{q^2}{2}\right)exp\left(\alpha\left(\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S\right)\frac{\vec{q}}{2}\right) \tag{D.39}$$

$$\times\int_0^{2\pi} d\theta\; exp\left(\alpha\,|\hat{\boldsymbol{\beta}}-\hat{\boldsymbol{\omega}}_S|\sqrt{1-\frac{q^2}{4}}\; cos(\theta)\right) \tag{D.40}$$

Now, I simplify using the following definition, where I_0 is the modified Bessel function of the first kind.

$$I_0(x) = \int_0^{2\pi} d\phi\; exp(x\; cos(\phi)) \tag{D.41}$$

Using this rule, as I have a remaining portion of G_4 of the form $\int_0^{2\pi} d\phi\, exp\,(x\, cos\,(\phi))$, I continue to simplify.

$$\begin{aligned}
G_4\left(\vec{\boldsymbol{q}}, \hat{\boldsymbol{\beta}}\right) &= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp\,(\alpha\,(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S))\, \frac{H\,(2-q)}{q} exp\left(\alpha \frac{q^2}{2}\right) \\
&\quad \times \int_0^{2\pi} d\phi'\, exp\left(\alpha\left(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S\right)\sqrt{1-\frac{q^2}{4}}\, cos(\phi')\right) \qquad \text{(D.42)} \\
&= \frac{e^{-4\alpha}}{(\Delta s)^3}\, exp\,(\alpha\,(\vec{\boldsymbol{q}} \cdot \hat{\boldsymbol{\omega}}_S))\, \frac{H\,(2-q)}{q} exp\left(\alpha \frac{q^2}{2}\right) \\
&\quad \times 2\pi I_0\left(\alpha\left(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\omega}}_S\right)\sqrt{1-\frac{q^2}{4}}\right) \qquad \text{(D.43)}
\end{aligned}$$

Finally, I have obtained the simplified form of G_4, and thus an exact analytical solution for calculating the FPI-SW for paths of four segments. Unfortunately, as seen by the complexity of the derivation of G_4, I am unable to derive higher order terms of G. Instead, I numerically integrate G_4 to produce G_5 and G_6 using the recursive definition of G.