Optimal Multiple Importance Resampling

Optimal Spatial Reuse for Monte Carlo Light Transport Simulation

William Narchi

Delft University of Technology

TU Delft

Optimal Multiple Importance Resampling

Optimal Spatial Reuse for Monte Carlo Light Transport Simulation

by

William Narchi

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday July 5, 2024 at 10:00 AM.

Student number:	5046122	
Project duration:	December 4, 2023 – July 5, 2024	
Thesis committee:	Prof. dr. ir. E. Eisemann,	TU Delft, Thesis Advisor
	Prof. dr. ir. G. Migut,	TU Delft
	Ir. M. Molenaar,	TU Delft, Daily Co-Supervisor

Cover:	Recreation of the Veach multiple importance sampling test scene by Yining Karl Li
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Ray tracing has experienced increasing adoption in various spaces of computer graphics. The **ReSTIR (Reservoir-based Spatiotemporal Importance Resampling)** family of techniques has enabled several orders of magnitude speedups in light transport simulation algorithms which rely on ray tracing [3, 26, 24, 25].

We introduce an extension to **WRS (Weighted Reservoir Sampling)** [5, 10], a key component of ReSTIR, to reduce the occurrence of duplicate samples in multi-sample reservoirs. Further, we show how samples from multiple reservoirs can be combined in an MIS-style estimator as opposed to resampling from them. Lastly, we combine these two components to compute optimal weights for this estimator in a similar vein to **OMIS (Optimal Multiple Importance Sampling)** [22].

Our direct lighting proof of concept implementation demonstrates the efficacy of our WRS extension, lowering the variance of ReSTIR, particularly with difficult lighting arrangements. Further, our MIS-style estimator shows a significant improvement compared to ReSTIR. In problem domains where resampling produces a function value for integration, such as path tracing, this comes at no additional cost. Lastly however, optimal weights do not appear to be beneficial for our approach.

Preface

To say that working on this thesis was a challenge would be an understatement. The process of conducting the background research necessary to understand the fundamentals was arduous, yet immensely enjoyable. The inventiveness required to produce a piece of scientific work was significantly more challenging. One feels like an intellectual pendulum, swinging between the polarizing extremes of invigorating brainstorming and despair-inducing trances wherein one stares at the same paper, dumbfounded, for days on end. As tumultuous as the experience has been at times, it has also been immensely educational. I am more knowledgeable about not just computer graphics, but my own natural proclivities.

The first of several rounds of thanks goes to my supervisors: Professor Elmar Eisemann and (soon to be Dr.) Mathijs Molenaar. Mathijs I would like to especially thank for being responsive and offering practical advice in areas where I needed it most. To Elmar, I would like to express my sincere gratitude for your heartwarming support and faith in my abilities. From giving me an impromptu motivational speech when I lost faith in my original topic to the brainstorming discussions and off-topic conversations we had, your assurance has been a large part of the fuel behind my fire. When teaching students about perspective transformations and shadow maps, remember that any one of them might be much like myself: capable, passionate, but lacking in assurance and support. You and colleagues in the Computer Graphics and Visualisation group can be the support they need to achieve greatness; you need only listen to their grievances and extend the olive branch they need.

My next round goes to my family. To my father, thank you for your continued belief in my decisions and your encouragement to expand my horizons in life, both professional and personal. To my mother, a sentence in a preface cannot do justice to the example you have set and the pillar you have provided upon which to construct the person I am; I would not *be* were it not for you. To my siblings, let this be a testament that you can accomplish great things if you set your mind to them; learn from my accomplishments, but more importantly my struggles, and strive to be better, as I know you can be. To my late grandfather, thank you for instilling in me a fascination with computers; it is the unshakeable force that drives me to do great work.

Enumerating my friends and instances of their support would necessitate a literary work of its own. In lieu, I would like to thank all of you for your care and support. To those who provided me invaluable brainstorming, a relief when the struggles of life needed to be escaped, and a cradle to nurse my grievances and woes. To those I have known for only a few months and to those whom have been my companions for years or even a decade. One does not stand on their own, but on the pillars of their community.

Lastly, I dedicate an ode to the unsung heroes of this university, academia, and our society at large: educators and teachers. Though academia is a thankless field to begin with, the sub-domain of education is especially onerous. Science is only as capable as its scientists, who are only as capable as their educators.

Sincere thanks are owed to the authors of “A Gentle Introduction to ReSTIR: Path Reuse in Real-time”. Their work was detrimental in my understanding of resampling and thus to the realisation of this thesis. I am especially proud of the background section of this thesis for its potential as an educational tool and in many ways it should be seen as a companion to the authors’ work for any prospective learners of ReSTIR and GRIS theory.

TU Delft is home to many other such unsung heroes. Of particular note and whom I would like to thank personally are Stefan Hugtenburg, Taico Aerts, Professor Gosia Migut, and Professor Jan van Gemert. Your imprints have and will continue to shape generations of engineers and scientists. Do not underestimate the impact you have on students' lives and always remember that you are truly the best of us. You inspire me to practice the most human of all acts: helping others.

William Narchi
Delft, June 2024

Contents

Abstract i

Preface ii

1 Introduction 1

2 Background and Related Work 2
2.1 Monte Carlo Integration . . . 2
2.1.1 Mathematical Background . . . 2
2.1.2 Application in Rendering . . . 2
2.2 Importance Sampling . . . 4
2.2.1 Theory . . . 4
2.2.2 Practical Example . . . 6
2.3 Multiple Importance Sampling . . . 7
2.3.1 Theory . . . 7
2.3.2 Practical Example . . . 8
2.4 Optimal Multiple Importance Sampling . . . 11
2.4.1 Motivation . . . 11
2.4.2 Approach . . . 13
2.5 Resampled Importance Sampling . . . 15
2.5.1 Theory . . . 15
2.5.2 Practical Example . . . 18
2.6 ReSTIR: Reservoir-based Spatiotemporal Importance Resampling . . . 20
2.6.1 Motivation and Overview . . . 20
2.6.2 Weighted Reservoir Sampling . . . 20
2.6.3 Spatiotemporal Reuse . . . 21
2.6.4 ReSTIR for Direct Illumination . . . 22
2.6.5 Sample Confidence . . . 24
2.7 GRIS: Generalised Resampled Importance Sampling . . . 26
2.7.1 Motivation and Overview . . . 26
2.7.2 Reuse Across Domains . . . 26

3 Method 31
3.1 Overview . . . 31
3.2 Neighbour Selection . . . 31
3.3 Multiple Sample Generation . . . 32
3.3.1 The Duplicate Samples Problem . . . 32
3.3.2 Dividing Candidates Into Subsets . . . 32
3.4 Estimator and Optimal Weights . . . 33
3.4.1 Combining Samples . . . 33
3.4.2 Arbitrary Unbiased Contribution Weights . . . 34
3.4.3 Evaluating Estimators . . . 34

4 Implementation 35
4.1 Overview . . . 35
4.2 Ray Tracer Specifics . . . 35
4.3 Parameters . . . 35

5 Results and Discussion 38
5.1 Preliminaries . . . 38
5.1.1 Testing Environment . . . 38

5.1.2 Primary Test Scene . . . 38
5.1.3 Additional Test Scenes . . . 39
5.1.4 Chosen Parameters . . . 40
5.1.5 Error Metric . . . 40
5.1.6 False Colour Insets . . . 40
5.2 Similarity Heuristics . . . 41
5.3 Canonical Candidate Count . . . 42
5.4 Reservoir Size and Neighbour Count . . . 43
5.5 Convergence Behaviour . . . 46
5.6 Runtime Analysis . . . 47
5.7 R-MIS Weights . . . 48
5.8 WRS With Subset Division . . . 49

6 Conclusion **50**
6.1 Future Work . . . 50
6.1.1 Ablation Study With Subset Division WRS . . . 50
6.1.2 Accumulating Unbiased Contribution Weight Estimates . . . 50
6.1.3 Neighbour Selection . . . 50
6.1.4 Application to Scenarios With Non-Trivial Spatial Resampling . . . 51

References **52**

A ReSTIR+ Reservoir Combinations **54**

B Source Code **56**

C Full Images **57**
C.1 Similarity Heuristics . . . 57
C.1.1 Final Renders . . . 57
C.1.2 Alpha Vectors . . . 58
C.2 Canonical Candidate Count . . . 59
C.2.1 Final Renders . . . 59
C.2.2 Alpha Vectors . . . 60
C.3 MAPE Heatmaps . . . 61
C.3.1 1 Iteration . . . 62
C.3.2 5 Iterations . . . 67

1

Introduction

Light transport simulation is a cornerstone in the field of computer graphics. In order to create an accurate rendering of a three-dimensional scene, the paths which light follows starting from light sources and ending at the camera from which the scene is observed need to be modelled.

MC (Monte Carlo) integration is the current industrial and scientific standard for performing light transport simulation [37]. Individual random light paths are simulated and their results combined to produce a coherent image [30]. However, MC integration techniques are prone to variance - which manifests itself as noise in the produced images - due to the complexity of the integration domain. In any given scene, an innumerable quantity of light paths, with varying degrees of contribution to the final image, can be modelled.

A substantial proportion of light transportation research focuses on reducing this variance. Avenues of research include techniques for modelling light propagation such as **BDPT (Bi-directional Path Tracing)** [31, 16, 23, 14] and **MC-MC (Markov Chain Monte Carlo)** techniques [38, 34, 15, 9]; intelligent sample generation techniques [13, 18, 7]; and denoising images obtained with limited sample counts [33, 44].

A foundational technique underlying many of these techniques is **MIS (Multiple Importance Sampling)**, which allows for combining random samples arising from different sampling techniques [37]. MIS necessitates the computation of weights which are usually assumed to be non-negative, though removing this assumption and allowing for negative weights has been shown to yield possible improvements [22].

Asides from being prone to variance, MC integration techniques are usually incredibly time-consuming and thus have often been only widely adopted for offline rendering up until recently [6, 12]. With the advent of readily available **GPUs (Graphics Processing Units)** with specialised ray intersection test acceleration hardware [28, 41], much research has spawned within the domain of real-time rendering where resources are heavily constrained and sample counts are low; as few as a single sample per pixel. The **ReSTIR (Reservoir-based Spatiotemporal Importance Resampling)** family of techniques is one such set of approaches that tremendously reduces computation times by reusing samples across neighbouring pixels and previous frames, effectively amortizing the cost of MC integration spatially and temporally [3, 26, 24, 25, 43]. ReSTIR-based techniques build on the concept of **RIS (Resampled Importance Sampling)**, which allows for the combination of samples (potentially generated via multiple techniques as with MIS) selected from a pool of candidates, guided by a target function which determines the selection probability of each candidate [35].

Our work aims to explore the applicability of optimal MIS weights as presented by [22] to the combination of samples generated via RIS. This is used to spatially combine samples from different pixels in a similar vein to ReSTIR, though geared towards offline rendering. We use similarity heuristics to determine a constant set of neighbours to resample per pixel. We provide a simple extension to **WRS (Weighted Reservoir Sampling)** to prevent the generation of duplicate samples. Lastly, we define arbitrary unbiased contribution weights in lieu of intractable RIS probability density functions.

2

Background and Related Work

2.1. Monte Carlo Integration

2.1.1. Mathematical Background

Monte Carlo integration is a method for evaluating integrals, usually used for integrals which do not have a (known) closed form solution. Given an integral of the form Equation (2.1), it can be numerically approximated using an estimator of the form Equation (2.2).

$$F = \int_{\Omega} f(x)\, dx \tag{2.1}$$

$$F \approx \langle F \rangle = |\Omega| \frac{1}{N} \sum_{i=1}^{N} f(X_i) \tag{2.2}$$

N independent random samples X_i[1] are uniformly generated across the domain of integration Ω, averaged, and scaled by the size of the integration domain to obtain an estimate of the integral. The expected value of this estimator is the integral itself, i.e. $\mathbb{E}[\langle F \rangle] = F$.

Two important properties of estimators are their **bias** and **variance**. Bias is the degree to which the expected value of the estimator differs from the integral to be estimated; the estimator shown in Equation (2.2) is unbiased. Variance is the expected difference between a particular evaluation[2] of the estimator and its expected value. With a basic estimator as in Equation (2.2), variance can only be reduced by increasing the number of samples N.

2.1.2. Application in Rendering

In light transport simulation, we attempt to solve the **rendering equation**. This equation describes the relationship between outgoing light from a point on a surface as viewed from a particular angle (radiance) and the incoming light from all directions in the unit hemisphere centered on that point and oriented along the surface's normal (irradiance). Figure 2.1 visualises this.

[1] Capital letters are used to denote random variables, as opposed to traditional variables as in Equation (2.1).
[2] This is often referred to as a 'realisation' of the estimator in the literature.

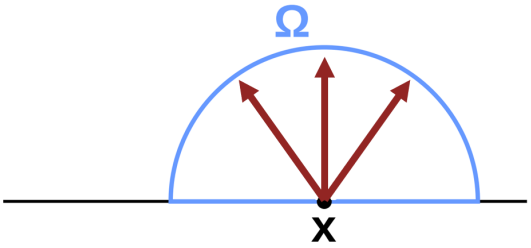


Figure 2.1: The rendering equation's integration hemisphere. Ω represents all possible light-carrying paths whose contribution we wish to evaluate. The arrows represent (uniformly distributed) directions that lead to discrete light samples. [21]

Direct Illumination

The first formulation of interest addresses direct illumination, which occurs as a result of light being emitted from a light source and reflecting off of a surface directly towards a viewer. Assuming a light path consisting of the sequence of vertices $\bar{x} = [x_0, x_1, x_2]$, where x_0 lies on a camera sensor (i.e. a pixel on the screen through which the scene is viewed), x_1 lies on a surface in the scene, and x_2 lies on a light source.[3] A possible formulation of this variant of the rendering equation is given by Equation (2.3).

$$L(x_1 \to x_0) = \int_{\mathcal{A}} f_s(x_2 \to x_1 \to x_0) G(x_1 \leftrightarrow x_2) V(x_1 \leftrightarrow x_2) L_e(x_2 \to x_1)\, dx_2 \tag{2.3}$$

The practical meanings of the presented terms are as follows:

- $L(x_1 \to x_0)$: The outgoing light (radiance) from vertex x_1 to vertex x_0.
- $\int_{\mathcal{A}} ... dx_2$: In this formulation, $\mathcal{A}$ is the set of all emissive surfaces (i.e. surfaces which emit light) in the scene. This term corresponds to integrating over all the points in the scene which emit light.
- $f_s(x_2 \to x_1 \to x_0)$: Evaluation of the **BSDF (Bidirectional Scattering Distribution Function)** of the surface on which x_1 lies given this particular light path (see Section 2.3.2 for more details). A BSDF describes how incident light is reflected for a particular material. In essence, this defines the quantity and color of light reflected along the direction $[x_1, x_0]$ given a particular incident quantity and color of light along the direction $[x_2, x_1]$.
- $G(x_1 \leftrightarrow x_2)$: A catch-all term describing geometric properties of the path $[x_1, x_2]$. In practice, this is usually the reciprocal of the squared distance between the points x_1 and x_2, in accordance with the inverse-square law.
- $V(x_1 \leftrightarrow x_2)$: Visibility between the vertices x_1 and x_2, i.e. if there are any surfaces between the two vertices causing an obstruction in the path.
- $L_e(x_2 \to x_1)$: The amount of light emitted (radiance) by the emissive surface on which x_2 lies in the direction $[x_2, x_1]$.

Global Illumination

The second formulation of interest addresses global illumination, which occurs as a result of light being emitted from a light source and reflecting off of several surfaces in sequence before being finally reflected towards a viewer. We assume light paths consisting of D vertices where x_0 lies on a camera sensor and all other vertices x_j lie on a surface in the scene. A possible formulation of this variant of the rendering equation is given by Equation (2.4).

$$L(x_1 \to x_0) = \sum_{D=2}^{\infty} \int_{\mathcal{B}^{D-1}} \left(\prod_{j=1}^{D-1} f_s(x_{j+1} \to x_j \to x_{j-1}) G(x_j \leftrightarrow x_{j+1}) V(x_j \leftrightarrow x_{j+1}) \right) L_e(x_D \to x_{D-1})\, dx_2 ... dx_D \tag{2.4}$$

[3] Path vertices should not be confused with mesh vertices. Path vertices can be any point on any surface in the scene. Mesh vertices denote the vertices which make up a three-dimensional model.

The practical meanings of the presented terms which differ from those outlined earlier are as follows:

- $\sum_{D=2}^{\infty}$: Summation of the contribution from all possible paths with lengths in the range $[2, \infty)$.
- $\int_{\mathcal{B}^{D-1}} ... dx_2 ... dx_D$: Integration over the product space of all the surfaces in the scene. $\mathcal{B}$ is the set of all surfaces in the scene. Effectively, this accounts for the fact that light can bounce off of any point in the scene towards any other point.
- $\left(\prod_{j=1}^{D-1} ...\right)$: This accounts for visibility, geometric properties, and BSDF evaluations at vertices along the path. In effect, how light is attenuated as it travels along the path and interacts with the surfaces at its constituent vertices.

Closed form solutions for the presented integrals are not known, and hence they are evaluated in practice via Monte Carlo integration. Discrete samples are taken consisting of light-carrying paths formed by vertices located in the scene. The paths are formed by tracing rays in the scene, evaluating the integral's terms at each point, and combining the resultant paths with Equation (2.2).

2.2. Importance Sampling

2.2.1. Theory

Motivation

It is often useful to draw samples from a distribution that better matches the function whose integral we wish to evaluate, as opposed to uniformly sampling the integration domain. This approach is known as importance sampling.

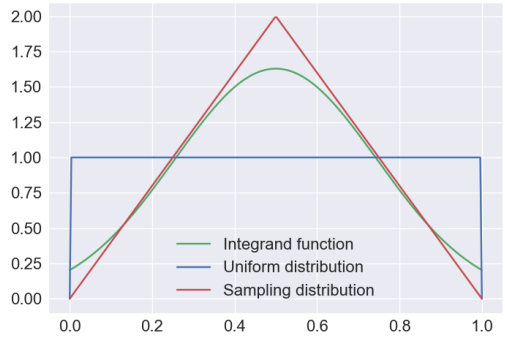
To visualise the benefit of this approach, consider Figure 2.2a. We would like to use MC integration to compute the integral of the green function 'Integrand function'. Ideally, we would like a higher concentration of samples around the central 'hump' of the function where its value is high, and fewer samples at the left and right 'tails' where its value is low. This would allow us to better represent the high value portions of the function which contribute most significantly to the integrand.

Uniformly drawing samples (according to the blue 'Uniform distribution') does not accomplish this. However, if our samples are instead drawn from the red 'Sampling distribution', we end up with a higher concentration of samples around the integrand function's central hump as can be seen in Figure 2.2b. The more closely a distribution matches the shape of the function to be integrated, the lower the variance of an estimator which uses this distribution to draw samples.

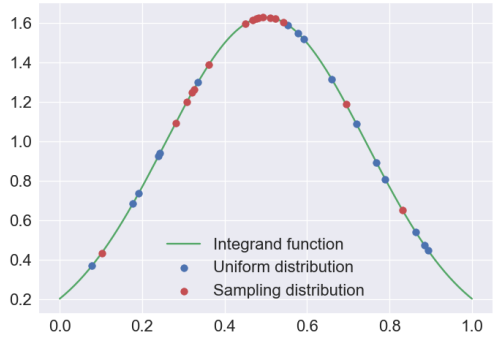
Figure 2.2c demonstrates this reduction in variance for the presented scenario. It illustrates how the **MSE (Mean Squared Error)** of the MC integral estimate varies with increasing sample counts for each distribution. MSE is a measure of the difference between ground truth and predicted values.

We compute MSE between the estimated MC integral $\langle F \rangle$ and the ground truth integral value F[4] as $(\langle F \rangle - F)^2$. It can be seen that the estimates based on the (red) 'Sampling distribution' have consistently lower MSE values than the estimates based on the (blue) 'Uniform distribution' for the same number of samples.

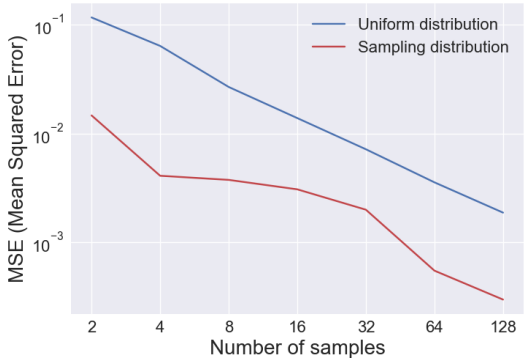
[4] In this example, the ground truth value F can be computed because the integrand function is a Gaussian distribution whose integral can be calculated in closed form. In light transport simulation, this ground truth is usually an estimate computed using an extremely large number of samples.



(a) A problem scenario where it is beneficial to apply importance sampling. It can be seen that the 'Sampling distribution' better matches the shape of the 'Integrand function' whose integral we would like to evaluate via MC integration.



(b) Comparison of drawing 16 samples from each of the two available distributions. Samples drawn from the 'Sampling distribution' are more concentrated around the central 'hump' of the distribution whereas samples drawn from the 'Uniform distribution' cover the integration domain more evenly.



(c) Comparison of MC integral estimates based on the presented distributions. The estimates based on the 'Sampling distribution' converge more quickly to the true integral value than those based on the 'Uniform distribution'. Note that the axes are logarithmic.

Figure 2.2: Motivating example for the benefit of importance sampling.

Estimator Specifics

An estimator that makes use of importance sampling has the form given by Equation (2.5)[5], where N is the total number of distributions being sampled from, n_i is the number of samples drawn from distribution i, and $p_i(X_{ij})$ is the probability of drawing the sample X_{ij} from the probability distribution p_i (i.e. the distribution's **PDF (Probability Density Function)**).

Each sample's contribution is weighted by its probability, such that samples with a higher probability of being generated are assigned a lower weight. This allows the estimator to remain unbiased [4]. Intuitively, this weighting prevents portions where the sampling density is high from over-contributing to the MC integral, biasing it towards the values that the integrand function assumes in regions of high sampling density.

$$\langle F \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{f(X_{ij})}{p_i(X_{ij})} \tag{2.5}$$

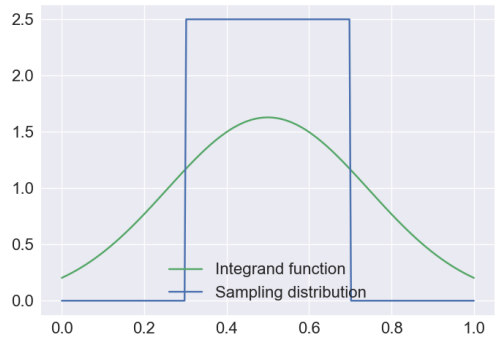
Supports

An important condition for guaranteeing the unbiasedness of Equation (2.5) is that the support of the distributions used to generate samples must cover the support of the function to be integrated. The support of a function f is the parts where the function is non-zero. The support of a random variable

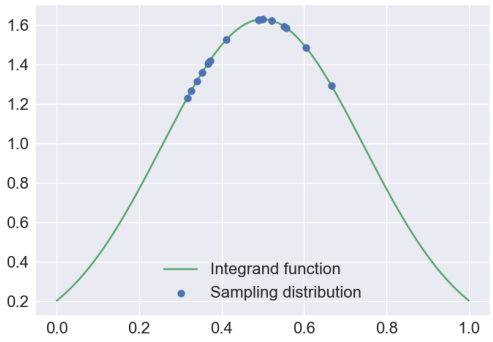
[5]This reduces to the basic estimator of the form Equation (2.2) if only a single sampling distribution that is uniform across the domain of integration is used. All samples thus have the same probability of being drawn $\forall i(p_i(X_i) = \frac{1}{|\Omega|})$ and $N = 1$.

X is the values it can take; for a continuous random variable, these are the values it can take with a positive probability. Thus, if X has PDF p, then $supp(X) = supp(p)$.

To illustrate, consider Figure 2.3a. We would like to evaluate the MC integral of the green 'Integrand function' f using samples drawn from the blue 'Sampling distribution' p. However, there are regions where f is non-zero and p is zero; these regions will not be represented in our estimator as no samples that cover them will ever be generated as can be seen in Figure 2.3b. Thus, the portion of the integrand covered by those regions will never be captured and our estimator will be biased.



(a) The integrand function and the distribution used to draw samples.



(b) 16 samples drawn from the distribution.

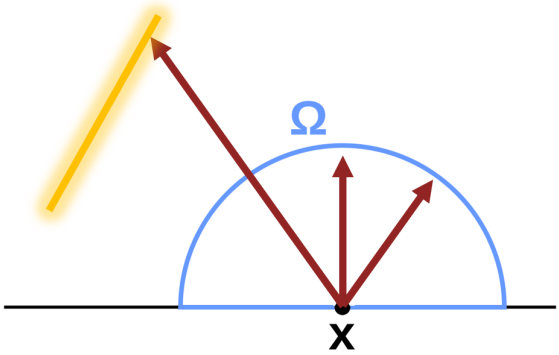
Figure 2.3: Example of a sampling distribution that does not cover the full support of the integrand function. Drawing samples from only this distribution does not allow us to cover the full domain of the integrand function that we need to integrate over.

2.2.2. Practical Example

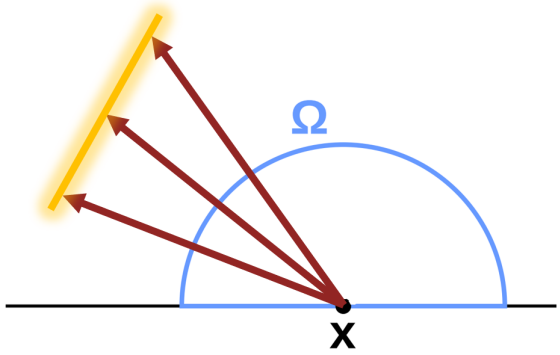
To illustrate how importance sampling can be practically used for light transport simulation, consider the scene depicted in Figure 2.4. Our scene contains a single yellow area light source and we wish to evaluate the direct illumination integral (Equation (2.3)) using MC integration.

Assuming we evaluate three samples, if we were to uniformly distribute the samples as in Figure 2.4a, two of our three samples would be useless as they do not lead to a light source. However, we can choose instead to importance sample in directions which only lead to light sources, yielding three useful samples as in Figure 2.4b. Effectively, we are importance sampling according to the $L(x_1 \to x_0)$ term of the integral.

This choice of sampling distribution is sound, as only directions which lead to light sources can contribute to direct illumination. As such, the support of the integrand function is covered by the support of the sampling distribution.



(a) Uniform sampling of all directions.



(b) Importance sampling directions leading to light sources.

Figure 2.4: Comparison of uniform sampling and importance sampling for direct illumination.

2.3. Multiple Importance Sampling

2.3.1. Theory

Motivation

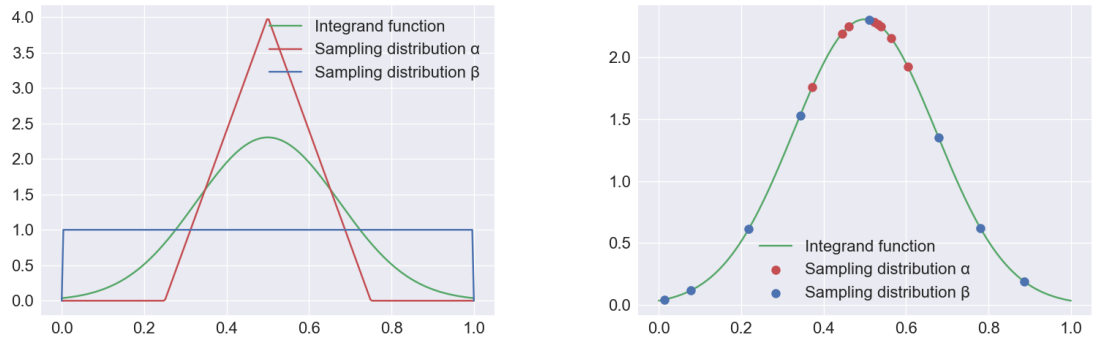
For integrands of more complex functions, a single distribution often does not capture the ranges of interest in its domain. As such, sampling from multiple distributions that fit different portions of the domain may be desirable.

Further, basic importance sampling assigns the same weight to each sample, namely $\frac{1}{N} \cdot \frac{1}{n_i}$ (see Equation (2.5)). This may not be desirable if the distributions being sampled from are particularly good at covering certain portions of the domain, but not others. If any one distribution does not match the integrand function well, the variance resulting from this mismatch is not mitigated. In effect, the variances of the distributions being sampled are added together if equal weights are assigned to samples as we effectively compute the average of separate estimators based on each of the distributions we draw samples from [43].

Lastly, if the support of *any* of the distributions used does not cover the full domain of the integrand function, the estimator will be biased. Portions not covered by the support of all distributions will be under-represented in the MC integral compared to the regions that *are*, leading to bias.

Visual Example

To illustrate, consider the situation in Figure 2.5a where we would like to evaluate the integral of the blue 'Integrand function' using MC integration. We have access to two sampling distributions α and β for generating samples. For the majority of the portion of the domain covered by α, we would like to prioritise samples generated by it (i.e. assign them a higher weight) as the distribution matches the shape of the integrand function quite well. Outside of α's domain however, samples from β should be prioritised as they allow us to cover the portions of the domain not covered by α. Not doing so would lead to bias as those regions would be under-represented as noted earlier. This is illustrated in Figure 2.5b where it can be seen that samples from α more effectively capture the high value central 'hump' of the integrand function, but only samples from β cover the two low value 'tails' of the integrand function.



(a) The integrand function and the two distributions used for drawing samples.

(b) 8 samples drawn from each distribution for a total of 16 samples.

Figure 2.5: Example of an integrand function whose domain is covered to different degrees of quality by two sampling functions. Distribution α is effective at covering the central 'hump', whereas distribution β equally covers the entire domain.

Estimator Specifics

First presented by Veach et al. [37], **MIS (Multiple Importance Sampling)** allows us to accomplish this by assigning each sample a weight, resulting in an estimator of the form Equation (2.6)[6] where $m_i(X_i)$ is the weight assigned to sample X_i.

In order for this estimator to remain unbiased, two conditions must be satisfied [37]:[7]

[6] This reduces to the importance sampling estimator of the form Equation (2.5) if all samples are assigned an equal weight $\frac{1}{N}$.

[7] Note the usage of lowercase x. This designates a concrete value that has been generated as opposed to a random variable.

- $\sum_{i=1}^{N} m_i(x) = 1$ whenever $f(x) \neq 0$
  The sum of utilised weights must be equal to one for all samples that generate a non-zero $f(x)$ (i.e. all samples within the support of f).
- $m_i(x) = 0$ whenever $p_i(x) = 0$
  If the probability of generating a particular value x from distribution p_i is 0 (i.e. x is not in the support of p_i), the weight assigned to this value must also be zero.

Combining these conditions allows for the MIS estimator to be unbiased if the union of the supports of the distributions covers the support of the integrand function, i.e. $supp(f) \subseteq \cup_{i=1}^{M} supp(p_i)$ where M is the number of distributions being utilised.

$$\langle F \rangle = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{m_i(X_{ij}) f(X_{ij})}{n_i p_i(X_{ij})} \tag{2.6}$$

Choosing Weights

A fundamental component in the success of an MIS estimator is the choice of weighting function. Common choices include:[8]

- **Balance heuristic**: $m_i(x) = \frac{p_i(x)}{\sum_{j=1}^{N} p_j(x)}$
  The most common choice in relevant applications and literature [43, 4]. It is the most optimal choice in terms of reducing variance provided that weights are not allowed to be negative (see Section 2.4) [22].
- **Power heuristic**: $m_i(x) = \frac{p_i(x)^\beta}{\sum_{j=1}^{N} p_j(x)^\beta}$
  A generalisation of the balance heuristic that can be more effective in scenarios where the sampling distributions quite closely match (portions of) the integrand function, i.e. have low variance. The relevant terms are raised to a power β, effectively 'sharpening' the weights. A value of $\beta = 2$ is commonly used [37].
- **Cutoff heuristic**: $m_i(x) = \begin{cases} 0 & \text{if } p_i(x) < \alpha p_{max} \\ \frac{p_i(x)}{\sum_{j=1}^{N} \{p_j(x) | p_j(x) \geq \alpha p_{max}\}} & \text{otherwise} \end{cases}$
  Where $\alpha \in [0, 1]$ and $p_{max} = \max_k p_k$. This effectively discards samples whose weights are sufficiently small, resulting in a similar sharpening effect as with the power heuristic, with more emphasis on reducing contribution from low weight samples.
- **Maximum heuristic**: $m_i(x) = \begin{cases} 1 & \text{if } p_i(x) = p_{max} \\ 0 & \text{otherwise} \end{cases}$
  Where $p_{max} = \max_k p_k$ as with the cutoff heuristic. The maximum heuristic effectively partitions the integration domain into separate regions that are covered by samples from only a single distribution each. It does not work as well as the other strategies in practice as, intuitively, too many samples are thrown away [37].

2.3.2. Practical Example

Bidirectional Scattering Distribution Functions

A **BSDF (Bidirectional Scattering Distribution Function)** describes how a material reflects and transmits incident light [30]. A BSDF combines the effects of a **BRDF (Bidirectional Reflectance Distribution Function)**, which describes how a material reflects light, and a **BTDF (Bidirectional Transmittance Distribution Function)**, which describes how a material transmits light through itself. We will focus on BRDFs in this section as they are the relevant component for our example.

BRDFs provide a mathematical description of how the energy of a single beam of light is distributed in all possible directions in the hemisphere around a surface point. Essentially, it describes how much light is reflected and in which directions it is reflected. Figure 2.6 shows the three general 'classes' of materials commonly referred to in light transport literature. n is the normal to the surface point x

[8] A visual comparison of the presented weights for a simple 1D integration problem can be found in Figure 2.12.

and v is a unit vector pointing towards a light source. r_v is a unit vector pointing in the direction of v's reflection with respect to x and n. The blue 'lobes' represent the quantity of light being reflected in different directions.

A 'Diffuse' surface reflects incident light uniformly in all directions, as shown by its lobe having a semi-circular shape. A 'Specular' surface reflects incident light only towards the direction of the reflection vector r_v. A 'Glossy' surface lies in between; a significant portion of incident light is reflected in the direction of the reflection vector r_v, though some light is reflected in other directions to varying extents.

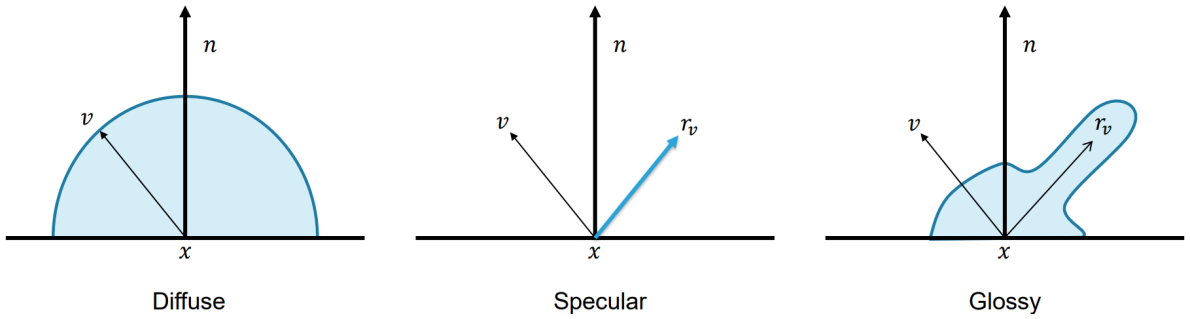
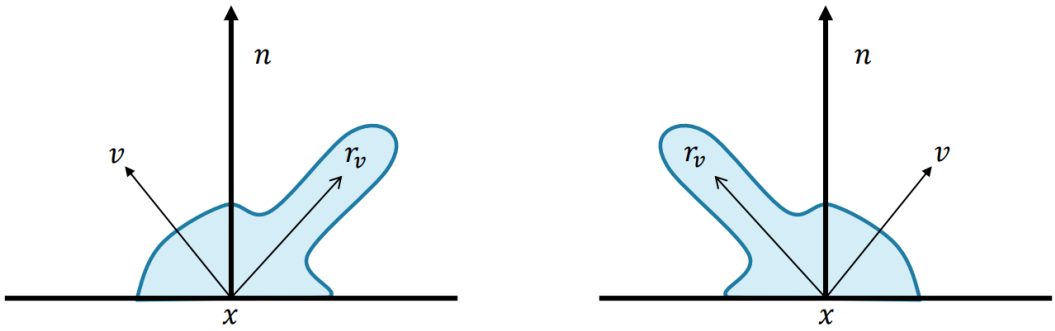


Figure 2.6: Three classes of BRDFs. They vary in the shape of the lobe indicating the directions where incident light is likely to be reflected. [20]

An important attribute of BRDFs is that they are symmetrical. That is, if the v and r_v vectors are reversed, the lobe is mirrored across the surface's normal. This is visualised in Figure 2.7.



(a) Original directions

(b) Reversed directions

Figure 2.7: Visual demonstration of the symmetric property of BRDFs. [20]

BRDF Sampling

We return to our direct illumination example from Section 2.2.2, but with a slight twist: the surface being examined is specular in nature. More specifically, it has the BRDF lobe depicted in Figure 2.8.

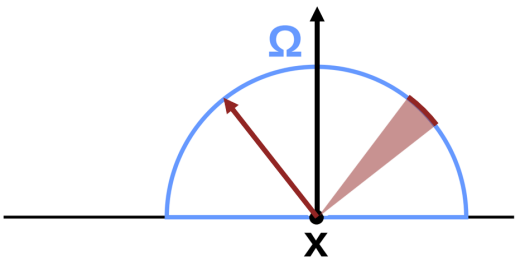
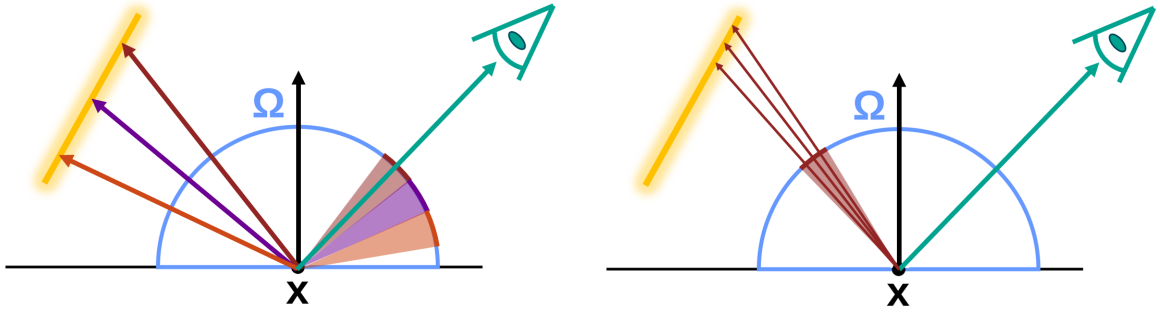


Figure 2.8: BRDF lobe of the surface being examined.

If we attempt to apply our prior approach of importance sampling directions leading to light sources, we

end up with the scenario depicted in Figure 2.9a. This would result in two of our three samples being wasted as the possible directions in which they could be reflected would not lead to our observer.

However, we can instead sample according to the surface's BRDF by leveraging the fact that BRDFs are symmetric and treating the direction leading to our observer as the v vector as defined earlier. This gives us a range of directions whose reflected energies have a non-zero probability of ending up at the observer, greatly increasing the chances of our samples contributing useful lighting information.



(a) Importance sampling directions leading to light sources. Each sample is depicted along with the lobe showing the distribution of directions where its energy is reflected.

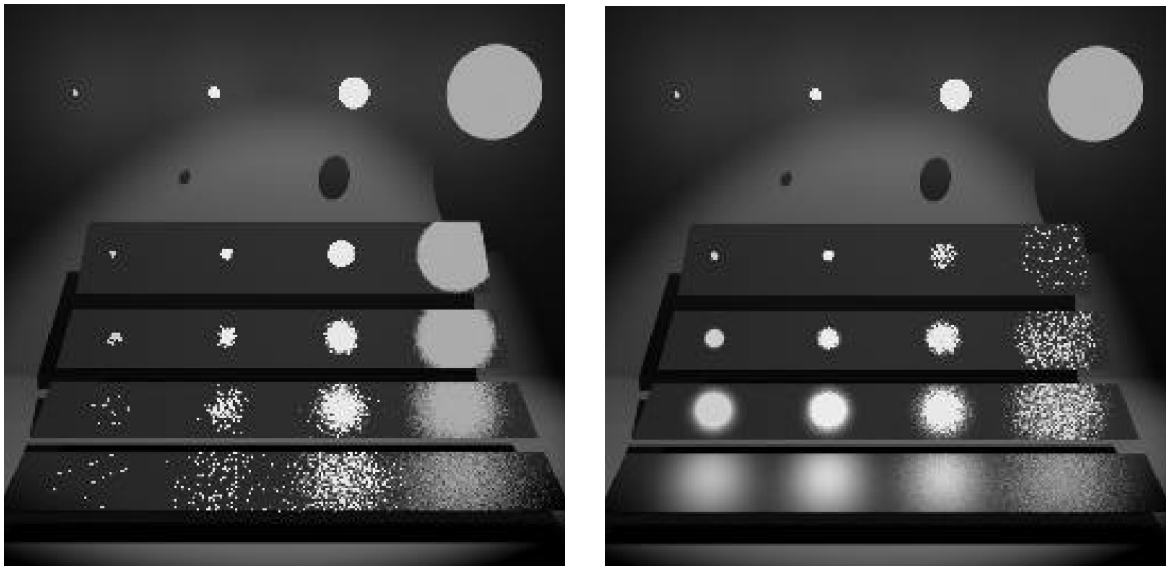
(b) Importance sampling directions according to the surface's BRDF.

Figure 2.9: Comparison of light sampling and BRDF sampling for direct illumination. An observer (teal eye) is also depicted.

Combining BRDF and Light Sampling

In the vast majority of scenes, we can rarely decide a priori which one of a given number of sampling techniques is optimal for particular scenarios. Consider once again the problem of estimating the direct illumination integral given access to a BSDF sampler and a light sampler.

Figure 2.10 depicts a scene with four spherical light sources of varying radii and color, plus a spotlight overhead. All spherical light sources emit the same total power. There are also four shiny rectangular plates, each one tilted so that we see the reflected light sources. The plates' BRDFs having varying levels of glossiness, which controls how sharp or fuzzy the reflections are. [37]



(a) Importance sampling directions according to the surface's BRDF.

(b) Importance sampling directions leading to light sources.

Figure 2.10: Comparison of light sampling and BRDF sampling for direct illumination in a 3D scene. The glossiness of the plates increases from the bottom to the top. [37]

Individually, each sampling technique is able to adequately cover only a portion of the integration do-

main. BRDF sampling performs well for glossy surfaces, whereas light sampling performs well for diffuse surfaces. Each sampling technique effectively samples according to a particular term of the integral, $f_s(x_2 \to x_1 \to x_0)$ for the BRDF sampler and $L_e(x_2 \to x_1)$ for the light sampler. We do not know a priori which term of the integral dominates across different portions of the integration domain. To address this, we can combine results from both sampling techniques using MIS, producing results that combine the strengths of both sampling techniques as can be seen in Figure 2.11.

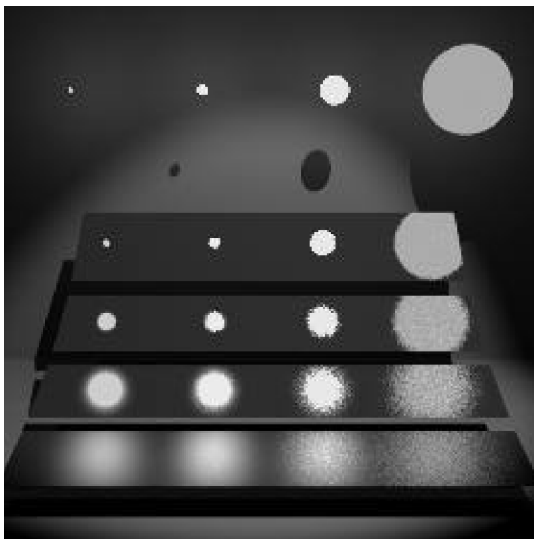


Figure 2.11: Combining samples produced by the BRDF and light samplers using MIS. [37]

2.4. Optimal Multiple Importance Sampling

The original formulation of MIS assumed weighting functions that did not produce negative weights, leading to the intuitive interpretation of MIS weights as a convex combination of sampling techniques. However, the conditions detailed in Section 2.3.1 do not necessarily forbid negative weights, allowing for an *affine* combination of sampling techniques as shown by Kondapaneni et al. [22].

2.4.1. Motivation

Visual Example

To demonstrate the potential benefits of negative weights, consider the 1D integration problems depicted in Figure 2.12. We wish to evaluate the integral of f via MC integration using an MIS estimator and we have access to three sampling distributions p_1, p_2, and p_3. One sample is taken from each technique.

In addition to the weighting schemes detailed in Section 2.3.1, we also consider a 'best-technique' scheme which assigns a weight of 1 to the distribution that best fits f (i.e. has the lowest variance relative to f) and a weight of 0 to all other distributions.[9] Lastly, we consider optimal weights with unconstrained (potentially negative) signs as produced by the technique of [22].

In the first row, the distribution p_2 very closely matches the shape of f. The balance heuristic (b) generally prioritises p_2, though it assigns larger weights to p_1 and p_3 relative to the tightly-fitting p_2 in portions of the integration domain, leading to higher variance in those portions. The power and cutoff heuristics (c-d) somewhat mitigate this due to their sharper weights being a better fit for this low variance scenario. The maximum heuristic (e) performs worse than the balance heuristic. The best-technique heuristic (f) produces significantly better results due to its selection of the tightly fitting p_2 as the only distribution of choice. Optimal weights (g) perform best due to the greater flexibility afforded by affine combinations; permitting negative weights allows for f to be more accurately modelled as a linear combination of the sampling distributions.

[9] Such a heuristic would be impossible to apply in practice as we almost never know the shape of the integrand function in advance.

In the second row, none of the distributions closely resemble the shape of f. The balance, power, cutoff, maximum, and best-technique heuristics (b-f) all perform similarly. This highlights a fundamental issue at the heart of these techniques: they represent attempts to determine which distribution best resembles f across different portions of the integration domain; this yields subpar results when none of the distributions are a good fit. Optimal weights (g) instead try to approximate f as a linear combination of the distributions, allowing for a much tighter fit.

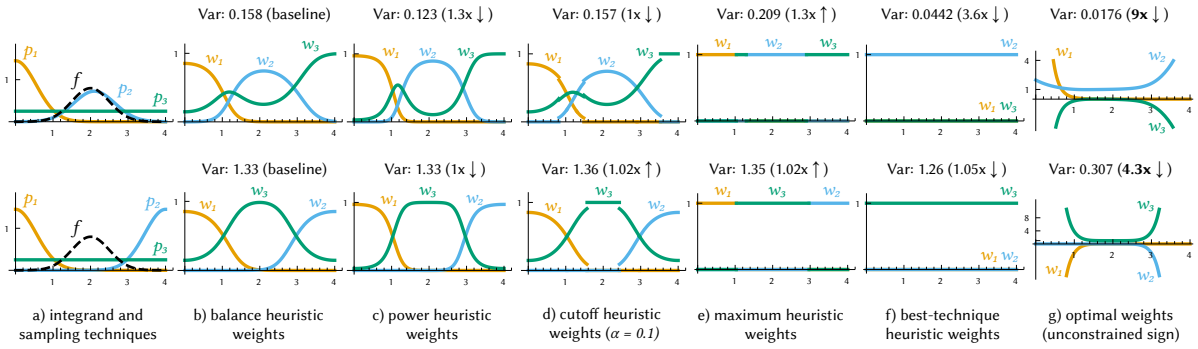


Figure 2.12: Comparison of various MIS weighting schemes for a 1D integration problem. The first column (a) shows the integrand function and the three available sampling techniques; the remaining columns (b-g) show the weights produced by each scheme. The top and bottom rows differ only in the shape of the p_2 distribution. [22]

MIS Estimator Variance Bounds

In the original derivation of the MIS estimator, an upper bound on its variance was explored [37]. Recall that an MIS estimator has the following form, where $w_i(X_i)$ is the weight assigned to sample X_i.[10]

$$\langle F \rangle = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{w_i(X_{ij}) f(X_{ij})}{n_i p_i(X_{ij})}$$

The variance of this estimator is given by Equation (2.7) [37]. To simplify notation, the inner product of two functions over a domain D is defined as $\langle a, b \rangle = \int_D a(x) b(x)\, dx$.

$$V\left[\langle F \rangle\right] = \underbrace{\sum_{i=1}^{N} \int_D \frac{w_i(x) f(x)}{n_i p_i(x)}\, dx}_{\text{first term}} - \underbrace{\sum_{i=1}^{N} \frac{1}{n_i} \langle w_i, f \rangle^2}_{\text{second term}} \tag{2.7}$$

The balance heuristic is the result of minimising the first term. The derived variance bound for an MIS estimator **utilising the balance heuristic** was constructed as the difference of the upper and lower bounds of the second term. The lower bound derivation does not make any specific assumptions, but in the upper bound derivation (Equation (2.8)), the second inequality $\star$ only holds if $\langle w_i, f \rangle \geq 0$. That is, only when the weights $w_i(x)$ are non-negative.[11] This suggests that the upper bound on the variance of the balance heuristic is in fact *larger* than suggested by the original proof from [37], motivating the benefits to be potentially reaped through the application of optimal weights with unconstrained signs. [22]

$$\sum_{i=1}^{N} \frac{1}{n_i} \langle w_i, f \rangle^2 \leq \frac{1}{\min_i n_i} \sum_{i=1}^{N} \langle w_i, f \rangle^2 \overset{\star}{\leq} \frac{1}{\min_i n_i} \left(\sum_{i=1}^{N} \langle w_i, f \rangle \right)^2 \tag{2.8}$$

[10] Note the change in notation from $m_i(X_{ij})$ to $w_i(X_{ij})$. This is done in order to maintain consistency with the notation of [22].

[11] In the context of rendering, the integrand function f is generally never negative as light is logically either present $f > 0$ or not $f = 0$. However, some approaches reformulate the rendering equation to allow for negative values [8].

2.4.2. Approach

Formal Definitions

Optimal weights can be formulated as the solution to the problem of minimising the variance of the MIS estimator (Equation (2.7)) in terms of the weights w_i while maintaining the conditions outlined in Section 2.3.1. To this end, we define a number of terms. Readers interested in the derivations leading to the terms defined in this section are advised to read Appendix B of [22].

Let f be a function to integrate within domain D, $\{p_1, ..., p_N\}$ a set of N sampling distributions whose union covers D, and n_i the number of samples drawn from p_i.

- **Technique matrix** $\mathbf{A}$: A symmetric $N \times N$ matrix with elements given by

$$a_{ik} = \left\langle p_i, \frac{p_k}{\sum_{j=1}^{N} n_j p_j} \right\rangle \tag{2.9}$$

- **Contribution vector** $\mathbf{b}$: A column vector of length N defined as $(b_1, ..., b_N)^\intercal$. Its elements represent contributions to the final integral $F = \int_D f(x)\,dx$ as the dot product $(n_1, ..., n_N) \cdot \mathbf{b}$ is equal to F. Its elements are given by

$$b_i = \left\langle f, \frac{p_i}{\sum_{j=1}^{N} n_j p_j} \right\rangle \tag{2.10}$$

Finally, we can define the column vector $\alpha = (\alpha_1, ..., \alpha_N)^\intercal$ as the solution to the system of linear equations

$$\mathbf{A}\alpha = \mathbf{b} \tag{2.11}$$

The optimal weights, denoted as w_i°, are then given by

$$w_i^\circ(x) = \alpha_i \frac{p_i(x)}{f(x)} + \frac{n_i p_i(x)}{\sum_{j=1}^{N} n_j p_j(x)} \left(1 - \frac{\sum_{j=1}^{N} \alpha_j p_j(x)}{f(x)}\right) \tag{2.12}$$

Practical Computation

The elements of $\mathbf{A}$ and $\mathbf{b}$ generally do not have a closed form solution and so they are estimated using MIS with the balance heuristic via the following estimators

$$\langle \mathbf{A} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \mathbf{W}_{ij} \mathbf{W}_{ij}^\intercal \quad \langle \mathbf{b} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{n_i} f(X_{ij}) S_{ij} \mathbf{W}_{ij} \tag{2.13}$$

Where S_{ij} is a scaling factor and $\mathbf{W}_{ij}$ is the column vector of the PDF of all sampling techniques evaluated at X_{ij} and scaled by S_{ij}. They are defined as follows

$$S_{ij} = \frac{1}{\sum_{k=1}^{N} n_k p_k(X_{ij})} \quad \mathbf{W}_{ij} = S_{ij}(p_1(X_{ij}), ..., p_N(X_{ij}))^\intercal \tag{2.14}$$

Finally, the estimator $\langle \alpha \rangle$ is computed via least squares minimization. This is because the *estimated* system $\langle \mathbf{A} \rangle \langle \alpha \rangle = \langle \mathbf{b} \rangle$ may be (close to) singular, hence making a direct solution infeasible. A consequence of this is that while the $\langle \mathbf{A} \rangle$ and $\langle \mathbf{b} \rangle$ estimators are unbiased, $\langle \alpha \rangle$ is generally *biased*. This bias in $\langle \alpha \rangle$ introduces *variance* in the final estimator $\langle F \rangle$.

Optimal MIS Estimator

Combining these findings into a practical estimator yields the **OMIS (Optimal Multiple Importance Sampling)** estimators depicted in Algorithm 1 and Algorithm 2. Differences between the two estimators are highlighted in red.

Algorithm 1 OMIS Progressive Estimator

$\langle \mathbf{A} \rangle \leftarrow 0^{N \times N}$; $\langle \mathbf{b} \rangle \leftarrow 0^{N \times 1}$; $\langle \alpha \rangle \leftarrow 0^{N \times 1}$; $result \leftarrow 0$
for $iterations \leftarrow 0$ **to** $maxIterations - 1$ **do**
  for $i \leftarrow 1$ **to** N **do**
    $\{X_{ij}\}_{j=1}^{n_i} \leftarrow$ draw n_i samples from distribution p_i
  end for
  if $(iteration \geq 1)$ **and** $iteration \mod U = 0$ **then**
    $\langle \alpha \rangle \leftarrow$ solve linear system $\langle \mathbf{A} \rangle \langle \alpha \rangle = \langle \mathbf{b} \rangle$
  end if
  $estimate \leftarrow$ evaluate $\langle F \rangle$ using $\langle \alpha \rangle$ ▷ Equation (2.6) and Equation (2.12)
  $result \leftarrow result + estimate$
  $\langle \mathbf{A} \rangle \leftarrow \langle \mathbf{A} \rangle + \sum_{i=1}^{N} \sum_{j=1}^{n_i} \mathbf{W}_{ij} \mathbf{W}_{ij}^{\intercal}$ ▷ Equation (2.13)
  $\langle \mathbf{b} \rangle \leftarrow \langle \mathbf{b} \rangle + \sum_{i=1}^{N} \sum_{j=1}^{n_i} f(X_{ij}) S_{ij} \mathbf{W}_{ij}$ ▷ Equation (2.13)
end for

return $result / maxIterations$

Algorithm 2 OMIS Direct Estimator

$\langle \mathbf{A} \rangle \leftarrow 0^{N \times N}$; $\langle \mathbf{b} \rangle \leftarrow 0^{N \times 1}$
for $iterations \leftarrow 0$ **to** $maxIterations - 1$ **do**
  for $i \leftarrow 1$ **to** N **do**
    $\{X_{ij}\}_{j=1}^{n_i} \leftarrow$ draw n_i samples from distribution p_i
  end for
  $\langle \mathbf{A} \rangle \leftarrow \langle \mathbf{A} \rangle + \sum_{i=1}^{N} \sum_{j=1}^{n_i} \mathbf{W}_{ij} \mathbf{W}_{ij}^{\intercal}$ ▷ Equation (2.13)
  $\langle \mathbf{b} \rangle \leftarrow \langle \mathbf{b} \rangle + \sum_{i=1}^{N} \sum_{j=1}^{n_i} f(X_{ij}) S_{ij} \mathbf{W}_{ij}$ ▷ Equation (2.13)
end for
$\langle \alpha \rangle \leftarrow$ solve linear system $\langle \mathbf{A} \rangle \langle \alpha \rangle = \langle \mathbf{b} \rangle$
return $\sum_{i=1}^{N} \langle \alpha_i \rangle$

The *Progressive* estimator repeatedly evaluates $\langle F \rangle$ using weights based on the most up-to-date $\langle \alpha \rangle$ estimate and averages the results of these evaluations. The $\langle \alpha \rangle$ estimate can be updated every U iterations rather than every single iteration as the linear system solution computation which updates it is relatively expensive. Thus, the *Progressive* estimator effectively averages the results of ever-improving estimates $\langle F \rangle$ of the integral F.

The *Direct* estimator instead exploits the definition of the α_i elements composing α. By definition [22]

$$\alpha_i = \int_D f(x) w_i^{\circ}(x)\, dx \tag{2.15}$$

As the weights sum up to 1 due to the MIS constraints (Section 2.3.1), the integral F can thus be expressed as the sum of the α_i terms

$$\int_D f(x)\, dx = \int_D f(x) \sum_{i=1}^{N} w_i^{\circ}(x)\, dx = \sum_{i=1}^{N} \alpha_i \tag{2.16}$$

However, this estimator will be generally *biased* as a result of the fact that the estimate $\langle \alpha \rangle$ is generally biased. In practice, the *Direct* estimator converges significantly quicker compared to the *Progressive* estimator. Lastly, the authors of [22] note that this bias was not significant in their experiments. Thus, this bias appears to not be empirically significant.

2.5. Resampled Importance Sampling

2.5.1. Theory

Motivation

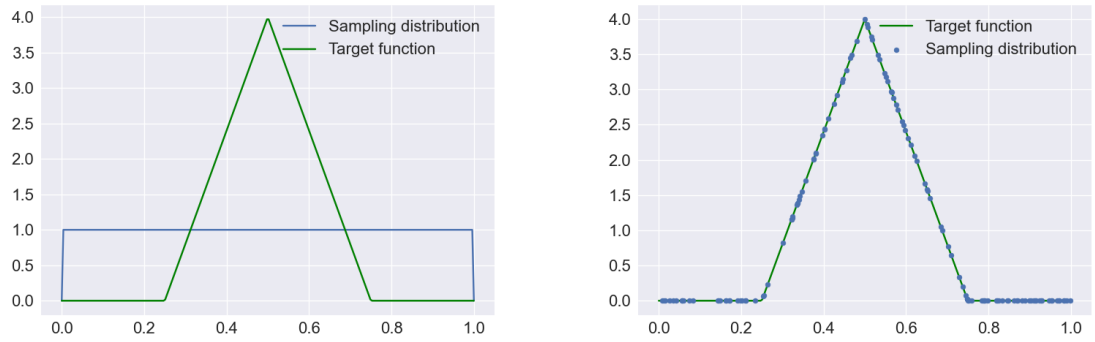
For many MC integration problems, it can be difficult to engineer a sampling distribution that matches the shape of the integrand function f, especially if f is the product of several individual terms. We may have access to distributions that can effectively capture each of the individual terms, but not their combined product.

Initially presented by Talbot et al. [35], **RIS (Resampled Importance Sampling)** accomplishes this by allowing us to weigh samples produced from a number of sampling-tractable distributions (i.e. distributions that we can actually draw samples from) according to a target function $\hat{q}$. In practice, $\hat{q}$ is an ordinary function which we *cannot* draw samples from **and can be unnormalised**. Samples are then stochastically selected based on their weight, such that samples with a higher weight have a higher probability of being selected.

Visual Example

To illustrate how RIS functions, we return to our example from Figure 2.2a but with an additional constraint: we can no longer draw samples from the green 'Sampling distribution'. However, we can *evaluate* the value of this function given a sample x.

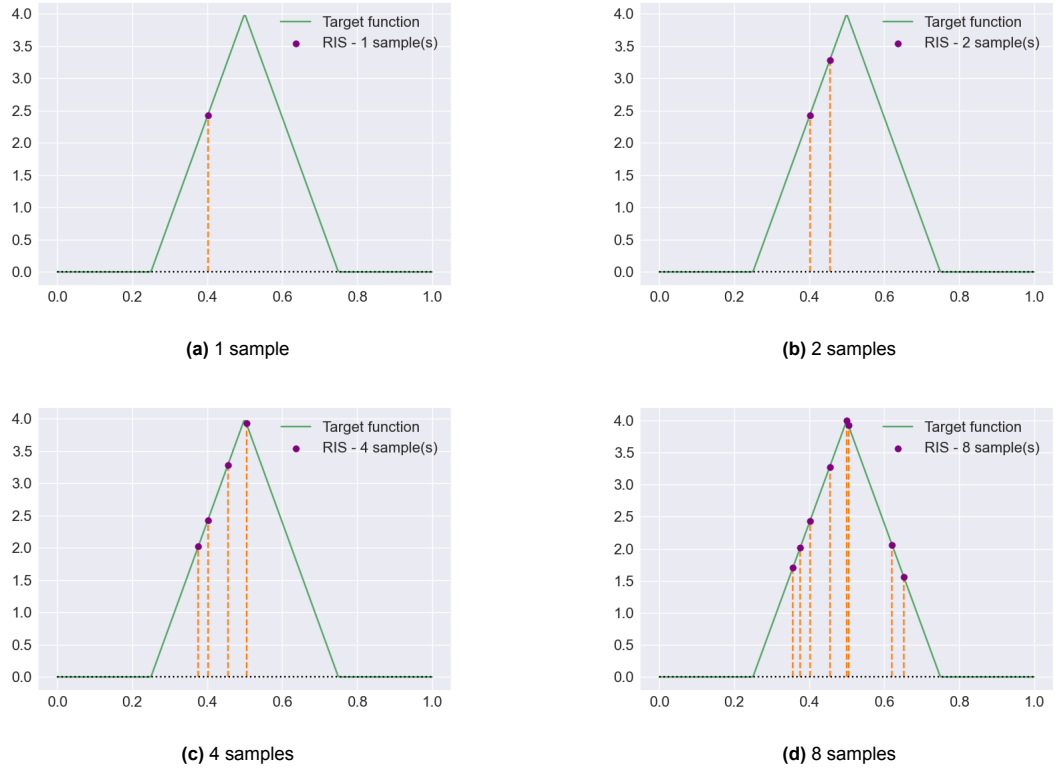
Figure 2.13 sets the scenario of our example once again. We begin by drawing 128 samples from the blue 'Sampling distribution', then we assign these samples weights based on their corresponding 'Target function' values and stochastically pick an increasing number of samples through RIS. The results of this process are shown in Figure 2.14. It can be seen that the distribution of the samples produced by RIS approaches the target function as the number of samples increases. Candidates which have a target function value of 0 are not present in the selected samples and samples are more concentrated around the central 'hump' of the target function.



(a) Distribution used to draw samples and target function used to assign weights for stochastic selection.

(b) 128 initial candidate samples drawn from the sampling distribution.

Figure 2.13: Motivating example for the benefit of RIS. We draw samples from the blue 'Sampling distribution' and then stochastically pick a subset using weights based on the green 'Target function'.



(a) 1 sample **(b)** 2 samples

(c) 4 samples **(d)** 8 samples

Figure 2.14: Samples drawn based on RIS. They are roughly distributed according to the target function, i.e. there is a larger concentration of samples around the central 'hump' of the target function. Orange dotted lines connect each sample to the x-axis to showcase portions of the domain where samples are concentrated.

RIS Estimator

An estimator that makes use of RIS has the form given by Equation (2.17). In order for the estimator to remain unbiased, the support of the target function $\hat{q}$ and the union of the supports of the distributions used to generate candidate samples must both cover the support of the integrand function.

$$\langle F \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{\hat{q}(X_i)} \cdot \sum_{j=1}^{M} \frac{1}{M} \cdot \frac{\hat{q}(X_j)}{p_i(X_j)} \tag{2.17}$$

Often in RIS-related work, the terms in Equation (2.17) are separately denoted as in Equation (2.18). The name and significance of each term is as follows:

- **Resampling weight** w_j
  The resampling weight of each candidate sample.
- **Sum of weights** w_{sum}
  Sum of the resampling weights assigned to the generated candidates.
- **UCW (Unbiased Contribution Weight)** W_{X_i}
  This replaces the $\frac{1}{p(X)}$ weighting term as used in a standard importance sampling estimator (Equation (2.5)). Unlike $\frac{1}{p(X)}$ however, W_{X_i} is not a deterministic function of X_i, but rather an estimate whose expected value is $\frac{1}{p(X)}$, i.e. $\mathbb{E}[W_{X_i}] = \frac{1}{p(X_i)}$.

$$w_j = \frac{1}{M} \cdot \frac{\hat{q}(X_j)}{p_j(X_j)} \qquad w_{sum} = \sum_{j=1}^{M} w_j \qquad W_{X_i} = \frac{1}{\hat{q}(X_i)} w_{sum} \tag{2.18}$$

The full sequence of steps for utilising an RIS estimator is as follows:

1. Generate M candidate samples from the available probability distributions p_j.
2. Compute a resampling weight w_j for each candidate.
3. Select N samples *with replacement*[12] from the candidates with probabilities proportional to the resampling weights w_j.
4. Compute an unbiased contribution weight W_{X_i} for each selected sample.
5. Evaluate the estimator using Equation (2.17).

Multiple Sampling Distributions with MIS

In a similar vein to Section 2.3, we may have access to several distributions for generating candidates. We would like to mix samples from these distributions such that they are appropriately weighed based on the portions of the integration domain that they cover.

To accomplish this, we simply modify the resampling weights from Equation (2.17) and Equation (2.18) into the form in Equation (2.19), where $m_j(X_j)$ is an MIS weight.[13]

$$w_j = m_j(X_j)\frac{\hat{q}(X_j)}{p_j(X_j)} \tag{2.19}$$

We can utilise any of the weighting functions described earlier in Section 2.3.1. If we have access to the underlying distributions from which the candidates were generated (i.e. the p_j's), we can use the exact same formulations of any of the presented weighting functions.

However, there might arise a scenario where we do not have access to the underlying distributions, such as when samples are generated from repeated applications of RIS. In such cases, we often instead have access to the target functions $\hat{q}_i$ that were used to guide the selection of the samples. We can use those target functions as proxies for the PDFs of the distributions of those samples. Effectively, we substitute target functions $\hat{q}_i$ where PDFs p_i are needed in MIS weighting functions if we do not have access to underlying distributions.

Stratified RIS Estimator

A key fault of the RIS estimator is the stipulation that selected samples are drawn *with replacement* from the pool of candidate samples. This often results in duplicate samples, particularly in scenarios where there is a relatively small number of high weight samples. This can drastically reduce potential benefits from increasing sample count (for both candidates and final selected samples).

To address this shortcoming, an unbiased RIS estimator can be constructed which operates on independent subsets which together form the full pool of candidates. This estimator is given by Equation (2.20).

$$\langle F \rangle = \sum_{i=1}^{N} \frac{f(Y_i)}{\hat{q}(Y_i)} \cdot \sum_{j=1}^{n_i} w_{ij} \tag{2.20}$$

The full sequence of steps for utilising this variant of the RIS estimator is as follows:

1. Generate M candidate samples from the available probability distributions p_{ij}.
2. Compute a resampling weight w_{ij} for each candidate.[14]
3. Divide the proposals into N subsets of size n_i, where $\sum_{i=1}^{N} n_i = M$

[12] "With replacement" indicates that selecting a sample does not prevent it from being selected again. This is opposed to selecting *without replacement* where a sample can only be selected once; after a sample is selected, the probabilities of selecting other samples are adjusted to account for the absence of the selected values. This ensures that the sum of probabilities of the remaining samples adds up to 1. Constructing an unbiased RIS estimator that makes use of selection without replacement is an open question [35].

[13] This form collapses to the more basic resampling weight if equal weights $\frac{1}{M}$ are used for all samples.

[14] Following on the generalised form given in Equation (2.19), this term includes an MIS weight. This weight should be relative to *all* candidate samples, not just the ones allocated to a particular subset.

4. Select N samples, one from each subset, with probabilities proportional to the resampling weights w_{ij}.
5. Compute an unbiased contribution weight $W_{X_{ij}}$ for each selected sample.
6. Evaluate the estimator using Equation (2.20).

The most important difference to note is that the scope of the w_{sum} term from Equation (2.18) has now changed. Instead of being the sum of the weights of *all* candidate samples, this sum is now per subset.

2.5.2. Practical Example

To illustrate the potential benefits of RIS, we build upon the direct illumination example from Section 2.3.2. The scene is depicted in Figure 2.15a. There are two key differences compared to our prior example:

- The lighting arrangement is significantly more complex. We have four lights, A through D, which are at different distances from the surface point. Further, they have different emissivities (brightnesses) as indicated by the saturation of their colour; more saturated lights are brighter.
- The BRDF lobe is still primarily specular, though its arc is larger.

We can attempt variations of prior approaches and importance sample directions according to different individual terms of the direct illumination integral.

Importance sampling according to the distance to light sources, such that directions leading to closer lights are more likely to be sampled (effectively sampling according to the $G(x_1 \leftrightarrow x_2)$ term) would lead to the scenario depicted in Figure 2.15b. Most of our samples would be skewed towards light source A as it is the closest. However, due to the specular BRDF of the surface, the range of directions where light coming from A can be reflected would not lead to our observer. Thus, this sampling strategy is suboptimal.

Next, we attempt importance sampling according to the emissivity (brightness) of light sources, such that directions leading to brighter lights are more likely to be sampled (effectively sampling according to the $L_e(x_2 \rightarrow x_1)$ term). This would lead to the scenario depicted in Figure 2.15c. Most of our samples would be skewed towards light source B as it is the brightest. Once again, due to the specular BRDF of the surface, the range of directions where light coming from B can be reflected would not lead to our observer. This sampling strategy is *also* suboptimal.

For our last valiant effort, we attempt importance sampling according to the BRDF of the surface (as in Section 2.3.2) such that directions that would result in light being reflected towards our observer are more likely to be sampled (effectively sampling according to the $f_s(x_2 \rightarrow x_1 \rightarrow x_0)$ term). This would lead to the scenario depicted in Figure 2.15d. Most of our samples would be skewed towards light source C, whereas a few would end up at light source D or no light source at all. This is an improvement over our prior two attempts, but still suboptimal. Light source C is the furthest light source and is also quite dim, so it would likely not produce high value samples. Further, some of our samples do not end up at a light source at all. Ideally, we would like most of our samples to originate from light source D due to it being relatively close, reasonably bright, and in the range of the surface's BRDF lobe.

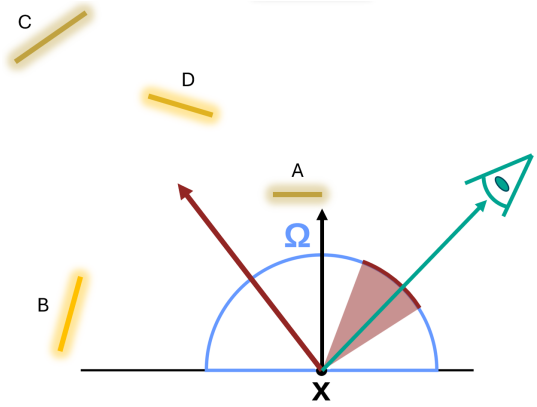
Our problem lies in the fact that the integrand function is a product of multiple terms. There usually exist techniques for importance sampling according to each of the individual terms, but not according to their product. RIS allows us to bypass this limitation. We can generate a number of candidate samples from any one of our sampling techniques (or multiple of them using MIS) and weigh them according to a target function which incorporates the *product* of these terms, such as[15]

$$\hat{q}(X_2) = f_s(X_2 \rightarrow x_1 \rightarrow x_0)G(x_1 \leftrightarrow X_2)L_e(X_2 \rightarrow x_1) \tag{2.21}$$

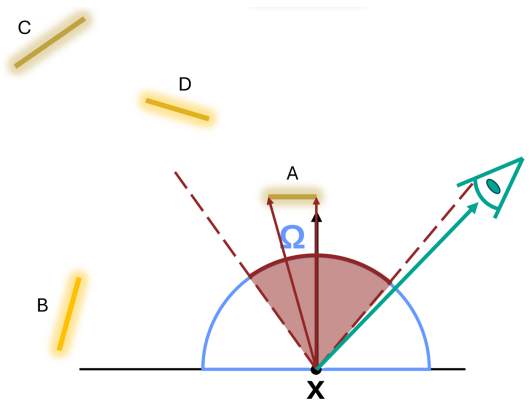
[15]Recall that light-carrying paths in the direct illumination problem are of the form $\bar{x} = [x_0, x_1, x_2]$. x_0 and x_1 are given, as they are the observer position and surface position respectively. Thus, we are trying to determine the ideal sample(s) for only x_2, hence why it is denoted with an uppercase X_2 to indicate that it is a random variable and *not* a known value.

This would produce the desired outcome, as samples which would result in no light being reflected to the observer as a result of the surface's BRDF would have a weight of 0 due to the $f_s(X_2 \to x_1 \to x_0)$ term. Further, the presence of the $G(x_1 \leftrightarrow X_2)$ and $L_e(X_2 \to x_1)$ terms means that samples from light sources that are closer and brighter, and thus more likely to have a larger contribution, are more likely to be selected.

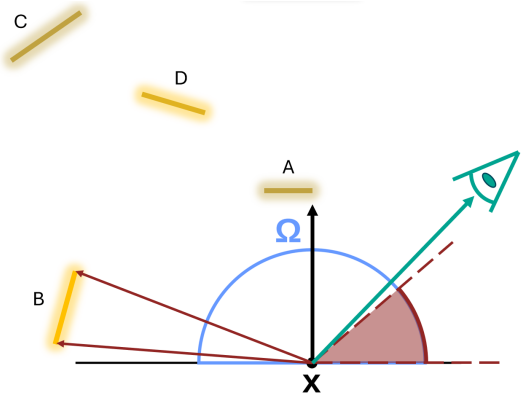
Note the absence of the visibility term $V(x_1 \leftrightarrow X_2)$ compared to the original direct illumination integrand function from Section 2.1.2. We omit this term as evaluating it would require tracing an expensive shadow ray to figure out if the light sample is occluded by objects in the scene. This means that our target function is not entirely optimal for the integral we are trying to estimate. However, this omission makes our application of RIS significantly more efficient. This approximation often yields good results in practice [3].



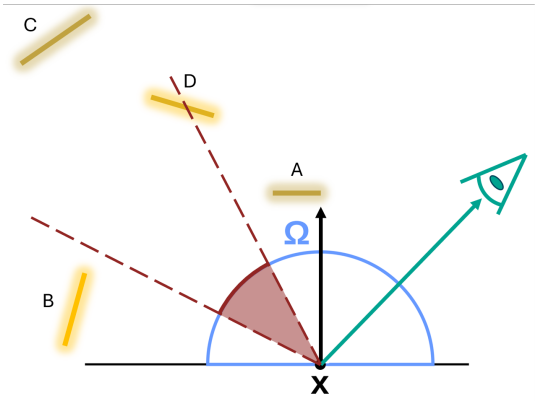
(a) The scene being examined. The surface's BRDF lobe, an observer, and four lights are depicted.



(b) Importance sampling according to the distance to light sources would skew samples towards A.



(c) Importance sampling according to the emissivity (brightness) of light sources would skew samples towards B.



(d) Importance sampling according to the BRDF of the surface would skew samples towards C. Note that in this example, the dotted lines represent the possible directions where light samples can be generated.

Figure 2.15: Comparison of various sampling techniques for direct illumination. Each technique on its own generates suboptimal samples. The dotted lines extend the bounds of relevant BRDF lobes to clarify all possible directions where light can be reflected given the light samples in each example.

2.6. ReSTIR: Reservoir-based Spatiotemporal Importance Resampling

2.6.1. Motivation and Overview

RIS provides a framework for generating samples that are distributed according to arbitrary target functions. This allows us to obtain samples whose distributions are better suited to the integration problem at hand, even if we are unable to directly generate samples from a distribution with the desired shape.

However, the number of samples needed to achieve a distribution that fits the target function can be tremendous. Even the simple problem presented in Figure 2.13 and Figure 2.14 requires a large number of candidates and final samples to achieve reasonable results. Generating candidates, performing weight computations, carrying out sample selection, and evaluating an estimator with the final samples can be computationally demanding.

To remedy this, the **ReSTIR (Reservoir-based Spatiotemporal Importance Resampling)** family of techniques builds upon RIS to amortize the costs of these computations spatially and temporally. This is accomplished by reusing chosen samples from neighbouring pixels and previous frames through repeated application of RIS. This effectively allows for a single pixel to select a set of final samples that represents an *enormous* set of initial candidates, resulting in potentially *extremely* high quality samples that capture high contribution segments of the integration domain.

2.6.2. Weighted Reservoir Sampling

Motivation and Overview

A major component of RIS is the generation of candidates, evaluation of resampling weights, and selection of a subset of samples from these candidates. The RIS procedure laid out in Section 2.5.1 requires that all M candidates and their weights are generated and stored upfront. This can lead to high storage costs, especially when such a process would have to be carried out repeatedly. This is often the case in rendering, as this process would be carried out for every pixel on a screen in parallel.

WRS (Weighted Reservoir Sampling) is a family of techniques that allow for N samples to be selected from a **stream** of M weighted candidates [5]. It does not require candidates to be stored; they are processed one-by-one as they arrive. Variants of WRS exist for selecting samples both with and without replacement, though we are only interested in the variants geared towards sampling *with* replacement to ensure unbiasedness when used with RIS [10].

Algorithm and Correctness

WRS maintains a *reservoir* of samples that are stochastically replaced when a new candidate arrives. It guarantees that each sample's probability of being selected is given by Equation (2.22), where w_i is the weight of candidate i. This leads to a data structure and update rule that correspond to Algorithm 3. Only the selected samples and a running sum of weights need to be stored.

$$P_i = \frac{w_i}{\sum_{j=1}^{M} w_j} \tag{2.22}$$

Algorithm 3 Weighted Reservoir Sampling

```
class Reservoir
    Y ← ∅                                   ▷ Set of N output samples
    w_sum ← 0                               ▷ Sum of weights
    function update(x_i, w_i)               ▷ New candidate and its weight
        w_sum ← w_sum + w_i
        for j ← 1 to N do
            if rand() < (w_i / w_sum) then[16]
                Y_j ← x_i
            end if
        end for
    end function
end class
```

Assume that m candidates have thus far been processed. When a new candidate x_{m+1} arrives, it can replace one of the stored samples with probability given by Equation (2.23). The probability of a previously selected sample x_i remaining in the reservoir is then given by Equation (2.24), maintaining the selection probability given by Equation (2.22)

$$\frac{w_{m+1}}{\sum_{j=1}^{m+1} w_j} \tag{2.23}$$

$$\underbrace{\frac{w_i}{\sum_{j=1}^{m} w_j}}_{\text{Initial choice}} \cdot \underbrace{\left(1 - \frac{w_{m+1}}{\sum_{j=1}^{m+1} w_j}\right)}_{\text{New candidate not selected}} = \frac{w_i}{\sum_{j=1}^{m+1} w_j} \tag{2.24}$$

2.6.3. Spatiotemporal Reuse

In the vast majority of rendering applications, neighbouring pixels are often trying to evaluate similar integrals. The points in the scene for which they estimate lighting integrals can lie on nearby objects, or even the same object. The points are also often surrounded by similar objects and are similarly spaced relative to light sources. As such, samples from one pixel are often quite good candidates for their spatial neighbours.

Similar correlations may exist temporally. In most rendering scenarios, scenes often do not change much from one frame to the next. In successive frames of an animated movie or real-time videogame, the positions of the camera, objects, and lights are relatively similar. As such, samples from the corresponding pixel in a previous frame are often good candidates for the current pixel. These correspondences are usually established via motion vectors, which describe the motion of the camera and objects in the scene. These motion vectors can then be used to figure out which pixel in the previous frame was observing the point that the current pixel is evaluating lighting for.

[16] `rand()` returns a uniformly distributed number between 0 and 1 inclusive (i.e. in the range [0, 1]).

This leads to the following typical workflow for ReSTIR-based algorithms, visualised in Figure 2.16:

1. **Initial candidate generation:** A number of candidates M is generated and a subset of N samples is drawn from them using RIS. This subset is approximately distributed according to the target function $\hat{q}$ used at this pixel. Samples generated in this step are often referred to as **canonical** samples.
2. **Temporal reuse:** The samples chosen by the corresponding pixel in the previous frame are resampled using RIS. Repeated application of ReSTIR with temporal reuse results in a continuously improving distribution of samples as good sample choices are propagated through time. Additionally, these choices can spread to *other pixels* in combination with spatial reuse.
3. **Spatial reuse:** A number of spatial neighbours k are identified in a small neighbourhood around the pixel. Samples from those pixels are used as candidates for further rounds of RIS. Several rounds of spatial reuse can be carried out.

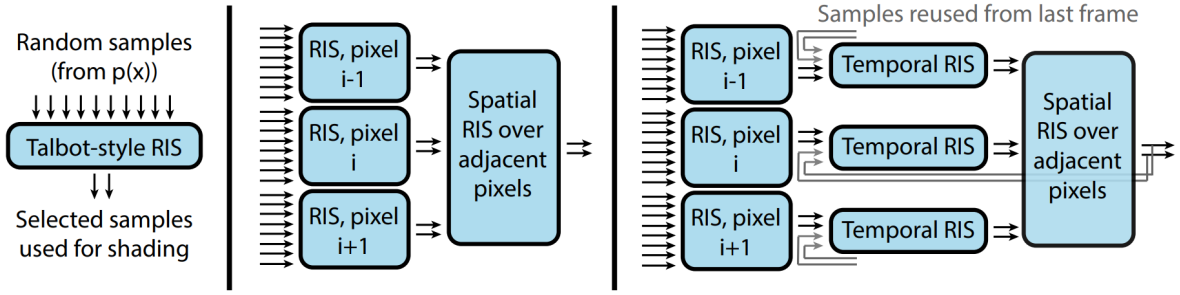


Figure 2.16: Typical workflow of ReSTIR-based algorithms [3].
(Left) The original conception of RIS by Talbot et al. selects a subset of samples from a set of candidates [35]. (Middle) RIS can be used to combine candidates from spatial neighbours. After an initial RIS round is done at each individual pixel, each pixel's chosen samples are used as candidates for further rounds of RIS. (Right) This methodology can be applied temporally, allowing for samples from prior frames to be used as candidates.

2.6.4. ReSTIR for Direct Illumination

Overview

To showcase the potential benefits of ReSTIR and illustrate its usage, we return to our direct illumination example from Section 2.3.2. This is the problem scenario addressed in the first paper within the ReSTIR family, presented by Bitterli et al. [3], which introduced the core ReSTIR methodology. Similarly to Section 2.5.2, we apply RIS to generate candidates and then select final samples, but we also reuse candidates spatiotemporally. Our example will use the following parameters:

- Initial candidates M: 32
- Final candidate(s) N: 4
- Number of spatial resampling rounds n: 2
- Number of spatial neighbours sampled per round k: 5

In the RIS example, we needed to consider only a single target function $\hat{q}$. This is because we only needed to consider a single pixel. Consider once again the representation of a light-carrying path in the direct illumination problem $\bar{x} = [x_0, x_1, x_2]$. For different pixels, the points x_0 and x_1 differ as demonstrated in Figure 2.17. Thus, each pixel i has its own target function $\hat{q}_i$. Note the omission of the visibility term compared to the original direct illumination integrand function, similarly to Section 2.5.2.

$$\hat{q}_i(X_2) = f_s(X_2 \to x_{i_1} \to x_{i_0}) G(x_{i_1} \leftrightarrow X_2) L_e(X_2 \to x_{i_1}) \tag{2.25}$$

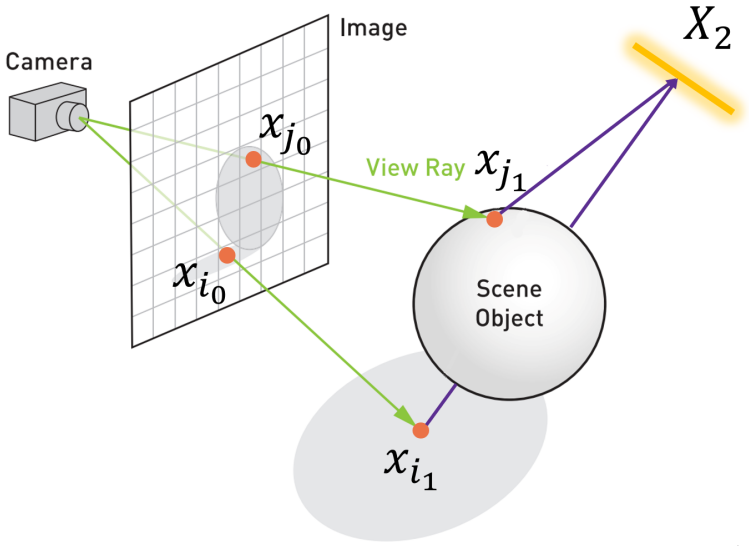


Figure 2.17: A simple scene containing one camera, one object, and one light. It can be seen that different pixels have different initial path segments $[x_0, x_1]$. [17]

Chaining RIS Passes

Recall from Section 2.5.1 that in order to carry out RIS at pixel j, we must compute a resampling weight

$$w_i = m_i(X_i)\frac{\hat{q}_j(X_i)}{p_i(X_i)}$$

This weight requires evaluating the PDF of the distribution that generated each sample. However, if we chain RIS as in ReSTIR, the distribution of samples generated by prior passes cannot be directly evaluated, as they are a result of the RIS process. However, recall that each sample that RIS outputs is associated with an unbiased contribution weight W_{X_i} whose expectation is equal to the reciprocal of the sample's PDF, i.e. $\mathbb{E}[W_{X_i}] = \frac{1}{p(X_i)}$. We can use this unbiased contribution weight in our chained RIS passes. Thus, our resampling weights under ReSTIR are of the form[17]

$$w_i = m_i(X_i)\hat{q}_j(X_i)W_{X_i} \tag{2.26}$$

For the MIS weights m_i, we can use any scheme that satisfies the requirements from Section 2.3.1.[18] However, as we do not have access to the source PDFs when combining samples from multiple pixels, we cannot utilise the PDFs in our weights as with standard MIS. We *do* have access to target functions $\hat{q}_i$, which we can use as a proxy for the PDFs, as these target functions are proportional to the PDFs.

For this example, we will use a variant of the balance heuristic, often referred to as the **generalised balance heuristic**. Assuming that we are selecting samples at pixel i and are resampling from D total pixels, this heuristic is of the form

$$m_i(x) = \frac{n_i\hat{q}_i(x)}{\sum_{j=1}^{D} n_j\hat{q}_j(x)} \tag{2.27}$$

Where n_i is the number of samples that pixel i provides. Since each pixel stores a fixed number of samples N (which is 4 in this example), the generalised balance heuristic can be simplified to the form

$$m_i(x) = \frac{\hat{q}_i(x)}{\sum_{j=1}^{D} \hat{q}_j(x)} \tag{2.28}$$

[17] If the actual PDF is known, such as during the initial sample generation step, we simply use the actual reciprocal of the PDF $\frac{1}{p_i(X_i)}$ as usual.

[18] MIS weights in ReSTIR are an involved topic. Their mishandling can lead to bias and there are several approaches with different caveats.

Effective Candidate Count

We now consider how the effective candidate count evolves after each step of ReSTIR. 'Effective candidate count' denotes how many candidates were considered in order to generate the final set of samples. This gives an indication of the quality of the final samples, as these samples have survived a resampling process where they were deemed the highest quality subset out of the considered candidates.

We assume that this is the second iteration of ReSTIR, and hence only one previous frame was rendered. We reverse the order of the spatial and temporal resampling steps to provide a more logical flow for how the effective candidate count progresses[19]

1. **Initial candidate generation:** $M = 32$ candidates are generated
   Effective candidate count $c_i = M = 32$
2. **Spatial reuse:** $n = 2$ rounds of reuse are carried out and $k = 5$ neighbours are resampled in a 30 pixel radius in each step
   Effective candidate count $c_s = c_i + (M \cdot n \cdot k) = 32 + (32 \cdot 2 \cdot 5) = 352$
3. **Temporal reuse:** A single temporal predecessor, which has already carried out the previous two steps, is resampled
   Effective candidate count $c_t = c_s + c_s = 352 + 352 = 704$

Lastly, if additional prior frames were rendered, the effective candidate count would grow even higher as a result of the temporal reuse step propagating the effective candidate count of each pixel forward in time.

2.6.5. Sample Confidence

Trusting High Confidence Samples

The process presented in the previous section has a key inefficiency: effective sample count is not represented in the resampling process. Put simply, we would like samples that were produced as a result of resampling many candidates to have a larger weight in the resampling process, as we are more confident in these samples' values due to the fact that they survived where many others did not.

To accomplish this, we can augment our MIS weights with **confidence weights**. We simply replace every instance of $\hat{q}_i(x)$ with $c_i\hat{q}_i(x)$, where c_i is the confidence weight of pixel i. Applied to the generalised balance heuristic, this yields

$$m_i(x) = \frac{c_i\hat{q}_i(x)}{\sum_{j=1}^{D} c_j\hat{q}_j(x)} \tag{2.29}$$

Practical Computation

To accurately determine confidence weights, we would have to carefully keep track of the number of unique candidates considered at each resampling step, and so determine the **effective sample count**. It is possible for duplicate candidates to be considered. Consider Figure 2.18, which visualises sample weights in a pixel grid during two successive frames:

A A high weight sample is generated at the center pixel.

B This sample is resampled by some of the pixel's spatial neighbours and stored. The sample's survival of this step increases its confidence weight.

C In the next frame, temporal resampling causes the sample to be retained. The sample's survival of this step further increases its confidence weight.

D Spatial resampling at the next frame causes the bright red pixels to resample the sample which they have already stored, resulting in duplicate candidates.

[19] This reversal is only for the purposes of this example. In practice, the temporal step is carried out first, followed by the spatial step.

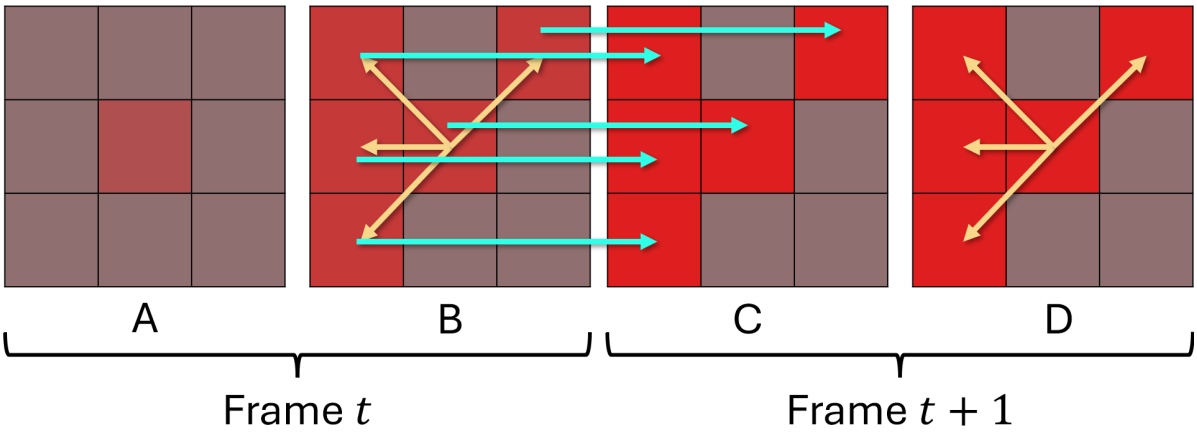


Figure 2.18: High weight samples can propagate, resulting in duplicate candidates. Golden arrows indicate spatial resampling. Teal arrows indicate temporal resampling. Higher saturation indicates higher sample weight.

In practice, the number of samples encountered by a reservoir is often used as a proxy for the effective sample count. Each sample generated in the initial candidate generation step is given a confidence of 1. Samples sourced from other pixels have a confidence weight equal to the number of samples encountered by their reservoir. This results in a slightly modified reservoir data structure and update rule, given by Algorithm 4. Differences to the original data structure are highlighted in red. Note that this confidence weight only represents an *upper bound*, as the actual confidence weight might be lower due to duplicate samples.

Algorithm 4 Weighted Reservoir Sampling w/ Confidence Weights

```
1:  class Reservoir
2:      Y ← ∅                                   ▷ Set of N output samples
3:      w_sum ← 0                               ▷ Sum of weights
4:      c ← 0                                   ▷ Confidence weight of output samples
5:      function update(x_i, w_i, c_i)          ▷ New candidate, its weight, and its confidence value
6:          w_sum ← w_sum + w_i
7:          c ← c + c_i
8:          for j ← 1 to N do
9:              if rand() < (w_i / w_sum) then
10:                 Y_j ← x_i
11:             end if
12:         end for
13:     end function
14: end class
```

Preventing Impoverishment

Figure 2.18 showcases another important potential pitfall in ReSTIR algorithms: **sample impoverishment**. This occurs as a result of high weight samples propagating and exponentially increasing their confidence weight. This prevents newly generated samples from being properly considered, as they have relatively much lower weight compared to these propagated samples.[20]

To combat this, confidence weights are often *capped* when performing temporal reuse. This cap is usually a multiple of the confidence weight of the reservoir carrying out the temporal reuse step, with 20 being a popular value. In ReSTIR literature, confidence weights are often denoted with the variable name M and capping their values is referred to as M*-capping*.

[20] Follow-up work has attempted to amend this more systematically using MC-MC mutations. [32]

2.7. GRIS: Generalised Resampled Importance Sampling

2.7.1. Motivation and Overview

The original conception of ReSTIR [3] adapted RIS to spatiotemporal reuse, and thus chaining RIS passes, in a relatively straightforward manner. However, the original conception of RIS was designed to work with fully independent candidates, which is often not the case with ReSTIR as duplicate candidates can arise (see Section 2.6.5). Moreover, traditional RIS assumes that all samples are in the same domain, such as points on light sources. If we wish to reuse samples across domains, such as reusing (parts of) paths through a scene, RIS cannot accommodate this.

GRIS (Generalised Resampled Importance Sampling) is an evolution of RIS, introduced by Lin et al., that generalises sample reuse [25]. It formalises the concept of **shift mappings** to enable sample reuse across domains, establishes conditions necessary to guarantee convergence, and provides bounds on variance.

2.7.2. Reuse Across Domains

Practical Example

Consider the global illumination integral from Section 2.1.2. We would like to evaluate this integral via MC integration and reuse samples via ReSTIR. Recall that a light-carrying path consisting of $D+1$ vertices in the global illumination problem is of the form $\bar{x} = [x_0, x_1, ..., x_D]$, where vertex x_0 lies on a camera sensor, vertex x_1 lies on an object in the scene, vertices $[x_2, ..., x_{D-1}]$ also lie on objects in the scene, and x_D lies on a light source. Light is emitted from the light sources, reflects off of objects in the scene, then finally reaches the observer.

Figure 2.19 depicts an example scene with two observers and a single light source. With global illumination, our samples consist of full *paths* as opposed to points on light sources are with direct illumination. It should be immediately clear that paths originating from one observer cannot be directly used in another observer. Their respective x_0 vertices will almost *always* be different, as observers are almost always in different positions. This is also often the case for x_1.

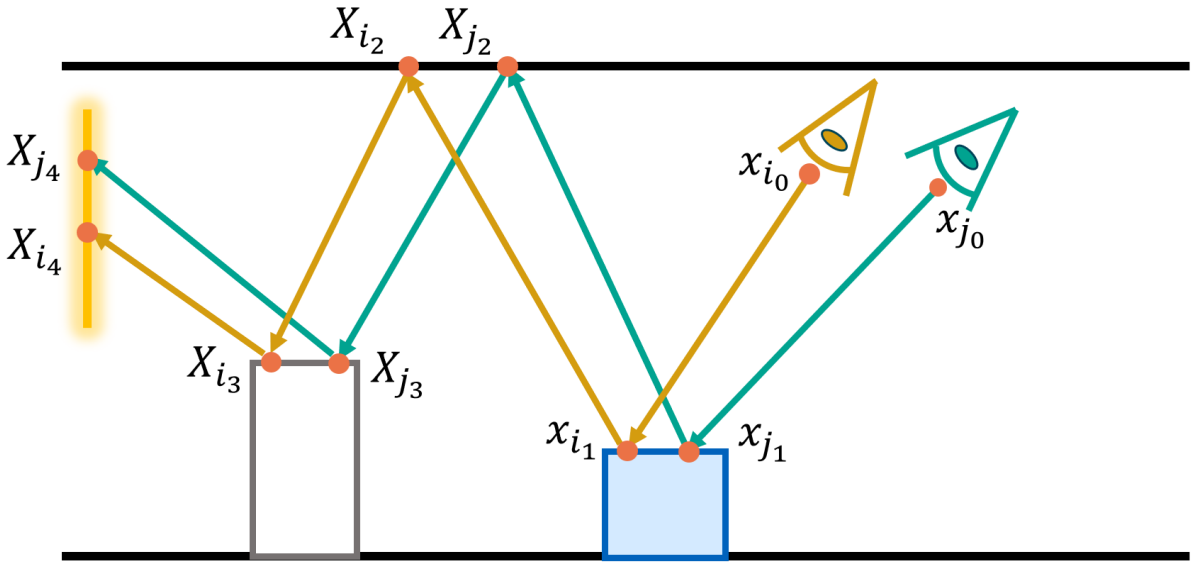


Figure 2.19: Illustration of the global illumination problem. Depicted are two observers (pixels), each evaluating a single sample consisting of a light-carrying path.

More formally, the **domain of integration** (the space of all possible samples) of two observers in the global illumination problem is not the same. Paths which originate from one observer cannot possibly originate from another. Fully reusing samples is not possible. However, at each vertex along a path, a light value is computed. At each vertex, we know how much light is arriving at that vertex. This information can be reused.

Shift Mappings

Shift mappings formally define how samples can be transformed from one domain to another. A sample is said to be 'shifted' from one domain to another. The term originates from gradient-domain rendering where shift mappings are used to determine horizontal and vertical pixel gradient estimates, which are then used with the initial path samples to construct the final image [19].

Given a source pixel i (a pixel whose path we want to shift), a target pixel j (a pixel we want to shift the path *to*), and a sample path X originating from pixel j, a sample Y originating from a shift mapping is formally defined as

$$Y = T_{i \to j}(X) \tag{2.30}$$

A shift mapping must be a bijective function from a subset of the source domain to all possible outputs of the function. More simply, it must have the following properties: [43]

- It is deterministic.
- A path from the source domain can map to *at most* one path from the target domain.
- No two paths from the source domain can map to a single path in the target domain. There must exist an inverse shift.
- The inverse shift must map back to the original path.
- Not all paths in the source domain need to be shiftable. A shift can *fail* on certain paths from the source domain, hence why shift mappings are bijective on only a *subset* of the source domain.

An example of a shift mapping known as the *reconnection shift* is depicted in Figure 2.20. This shift mapping simply connects paths at the third vertex. More formally, given a path $\bar{x}_i = [x_{i_0}, x_{i_1}, x_{i_2}, ..., x_{i_D}]$ in the source domain i and a path $\bar{x}_j = [x_{j_0}, x_{j_1}, x_{j_2}, ..., x_{j_D}]$ in the target domain j, the shifted path is defined as[21]

$$\bar{y} = T_{i \to j}(\bar{x}_i) = [x_{j_0}, x_{j_1}, x_{i_2}, ..., x_{i_D}] \tag{2.31}$$

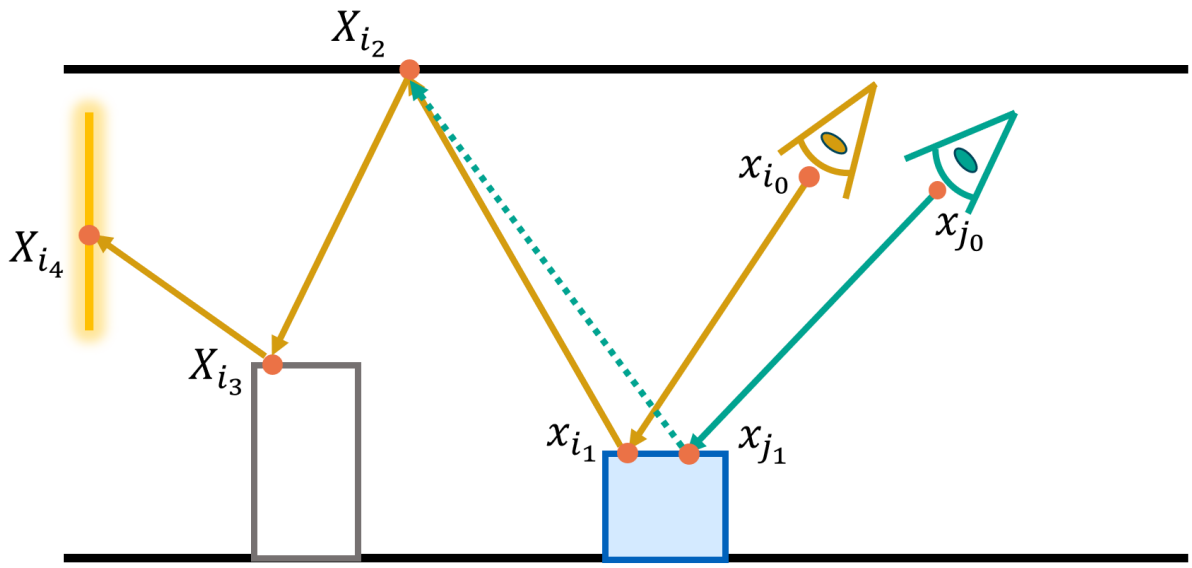


Figure 2.20: A potential shift mapping that allows for (partially) reusing samples across paths. This particular shift mapping is known as the *reconnection shift*.

[21] Vertices denoted in grey are fixed. They are already known as they are the positions of the observer and the point being observed.

This shift is not ideal if the third vertex on the source path lies on a glossy or specular surface. This is because the path segment $[x_{i_1}, x_{i_2}]$ in the original path is likely to carry a lot of light information, but the shifted path segment $[x_{j_1}, x_{i_2}]$ may not due to this direction being in a low output portion of the surface's BRDF lobe (see Section 2.3.2).

Luckily, a number of shift mappings exist in the relevant literature, such as the *random replay shift* and the *half-vector shift*, which address different concerns such as specular surfaces. Readers are advised to refer to section 3 of [19] and sections 6.3-6.6 of [43] for more information.

Jacobian Determinants

Consider the scenario depicted in Figure 2.21. It can be seen that the inputs are evenly distributed in the range $[1..5]$. The outputs however, are not evenly distributed in the range $[1..25]$. The relative distance between any two values changes when a non-trivial function is applied; it is scaled by a factor.

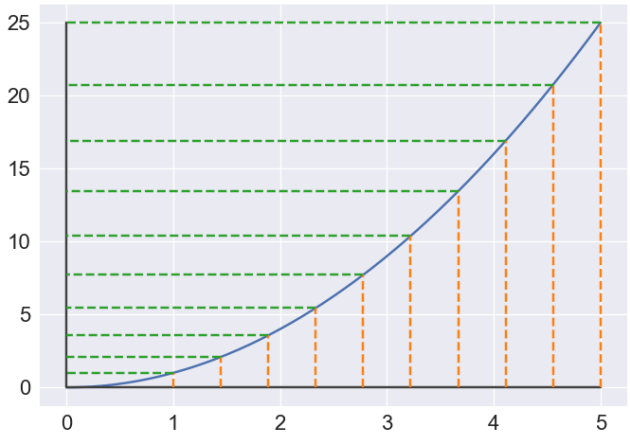


Figure 2.21: Mapping evenly spaced values in the range $[1..5]$ with the function $f(x) = x^2$. It can be seen that the distribution of the inputs (orange) is not the same as the distribution of the outputs (green).

This factor is the determinant of the Jacobian matrix (often referred to as just the 'Jacobian') of the function. Shift mappings, being functions, change the distribution of their inputs. As such, they need to be taken into account when resampling as they change the distribution of samples. Shift mappings in the literature almost always have a known geometric formula for their Jacobian determinant.

Given an input X and a shift mapping T, the determinant of the shift mapping's Jacobian for this input is denoted as $|T'(X)|$. If $Y = T(X)$ and X has PDF value given by $p_X(X)$, the PDF value of Y is given by

$$p_Y(Y) = \frac{p_X(X)}{|T'(X)|} \tag{2.32}$$

Analogously, given an unbiased contribution weight W_X (whose expected value is $\mathbb{E}[W_X] = \frac{1}{p_X(X)}$) for input X, the unbiased contribution weight of Y is given by

$$W_Y = W_X \, |T'(X)| \tag{2.33}$$

Using GRIS

Putting the information we have gathered thus far together, we have the following procedure for conducting a round of resampling with GRIS: [43]

1. Take M inputs $(X_1, ..., X_M)$, each from its own domain Ω_i.
2. Map the samples into the target domain Ω as $Y_i = T_i(X_i)$.
3. Evaluate MIS weights $m_i(Y_i)$ for all Y_i.
4. Evaluate resampling weights $w_i = m_i(Y_i)\hat{q}(Y_i)\underbrace{W_{X_i}\left|T_i'(X_i)\right|}_{W_{Y_i}}$ for all i.
5. Choose j samples Y_j randomly from the Y_i proportionally to w_i.
6. Evaluate an unbiased contribution weight for each selected sample $W_{Y_j} = \frac{1}{\hat{q}(Y_j)}\sum_{i=1}^{M} w_i$.

MIS Weights

Recall that MIS weights in ReSTIR usually make use of target functions $\hat{q}_i$ as proxies for PDFs at different pixels. Further, it is assumed that all samples for which target functions are evaluated originate from the same domain. An example is the simplified version of the generalised balance heuristic (Equation (2.28)). Given that we resample from D pixels in total, it is of the form:

$$m_i(x) = \frac{\hat{q}_i(x)}{\sum_{j=1}^{D}\hat{q}_j(x)}$$

In GRIS however, each sample $Y_i = T_i(X_i)$ originates from a source domain Ω_i and is mapped to the target domain Ω of the pixel conducting resampling. We are now faced with three problems:

1. In order to evaluate target functions $\hat{q}_i$ at other pixels, we must shift our newly obtained samples into the domains of those pixels.[22]
2. The target functions act as proxies for the PDFs in the *source* domains Ω_i; we require proxies in the target domain Ω.
3. Shift mappings can fail. We may not be able to successfully shift a sample from the target domain Ω to a source domain Ω_i.

For each pixel that we are resampling from, we define a shift mapping T_i that maps samples from that pixel to the target pixel conducting resampling. This shift mapping can be *inverted*, allowing us to map samples from the target pixel to the source pixel as $Z = T_i^{-1}(Y)$.[23] We can now evaluate the target function at the source domain Ω_i as $\hat{q}_i(Z)$. This resolves the first problem and gives us a proxy for the PDF at the source domain Ω_i, namely $\hat{q}_i(Z)$.

Building on this by using Equation (2.32), we can now obtain a proxy for the PDF at the target domain Ω as

$$\frac{\hat{q}_i(Z)}{|T_i'(Z)|} \tag{2.34}$$

Evaluating this form would require us to shift Z from the source domain Ω_i back to the target domain Ω in order to compute $|T_i'(Z)|$. However, we already know that this would give us our original sample Y as $T_i(Z) = T_i(T_i^{-1}(Y)) = Y$. As such, we can instead leverage the inverse function theorem and multiply by the Jacobian of the inverse shift [43]. This resolves the second problem and gives us a proxy for the PDF at the target domain Ω of the form

$$\hat{q}_i(Z)\left|T_i^{-1'}(Y)\right| \tag{2.35}$$

[22] It is important to stress that the sample being shifted here is Y_i, the sample belonging to the domain of the pixel conducting the resampling. We do not shift the *original* sample X_i.

[23] The stipulation that there must be an inverse shift associated with every valid shift mapping (Section 2.7.2) permits this.

Lastly, we can resolve our third problem by simply setting our proxy PDF to 0 if the inverse shift mapping fails. Combining these findings, we coin our proxy PDF as "$\hat{q}$ from i" and define it as: [25]

$$\hat{q}_{\leftarrow i}(Y) = \begin{cases} \hat{q}_i(Z) \left| T_i^{-1'}(Y) \right| & \text{if shift is successful} \\ 0 & \text{otherwise} \end{cases} \tag{2.36}$$

This grants us a generalised balance heuristic between different domains of the form

$$m_i(x) = \frac{\hat{q}_{\leftarrow i}(x)}{\sum_{j=1}^{D} \hat{q}_{\leftarrow j}(x)} \tag{2.37}$$

We can also include the confidence weights from Section 2.6.5, yielding the form

$$m_i(x) = \frac{c_i \hat{q}_{\leftarrow i}(x)}{\sum_{j=1}^{D} c_j \hat{q}_{\leftarrow j}(x)} \tag{2.38}$$

3

Method

3.1. Overview

The presented work outlines a method for combining samples across pixels using RIS in a similar vein to ReSTIR. However, rather than resampling from spatial neighbours, we use their final samples as-is in an MIS-style estimator.

Our method consists of the following sequence of steps, carried out for each pixel:

1. Select a fixed set of neighbours to resample from via similarity heuristics based on primary ray information.
2. Generate a number of canonical samples using RIS.
3. Combine canonical samples with resampled samples from neighbours, update optimal weight estimates, and evaluate the estimator, similarly to [22].

3.2. Neighbour Selection

In order to estimate optimal weights for combining samples across neighbours, the choice of neighbours to resample from should remain fixed. This allows us to build up these estimates over the course of several iterations of our approach.

Ideally, we wish to select neighbours with a higher likelihood of providing useful samples. More formally, these would be neighbours with a similar distribution of samples, i.e. ones whose target functions $\hat{q}_i$ produce similar results to the current pixel's target function $\hat{q}$ given the same set of samples.

To accomplish this, we use the following heuristics to classify neighbours in a small window around the current pixel as being either similar or dissimilar in a similar vein to [3] and similarly to edge-stopping functions for bilateral filters [11, 29]. These heuristics are based on the primary ray (i.e. the ray from the pixel to the point in the scene being observed) and corresponding standard hit information provided by ray tracers.

- **Geometry ID**: Valid neighbours must be observing the same object in the scene. A neighbour is considered dissimilar if the ID of the geometry it is observing is different.
- **Depth**: Pixels with large disparities in depth (i.e. how far away they are from the camera) are likely to have different lighting distributions. A neighbour is considered dissimilar if the difference between its depth and the depth of the current pixel is larger than a user-defined percentage of the current pixel's depth.
- **Surface normals:** Pixels with large disparities in surface normals are not likely to benefit from the same light samples. A neighbour is considered dissimilar if the angle between its normal and the normal of the current pixel is larger than a user-defined value.

We use a square window with a user-adjustable length to define the neighbourhood in which neighbours should be considered. Each neighbour is classified as either similar or dissimilar and then the desired number of resampling neighbours is picked at random from the set of similar neighbours. If an insufficient number of similar neighbours cannot be found, the deficit is made up for using dissimilar neighbours.

3.3. Multiple Sample Generation

3.3.1. The Duplicate Samples Problem

In ReSTIR and ReSTIR-derived approaches, a problem arises when reservoirs are set up to store several samples. They often end up storing duplicate samples. Figure 2.18 demonstrates an exemplary scenario. This problem is especially pronounced in (partially) shadowed regions where a small number of high value samples end up dominating due to there being few lights in the scene that can contribute to these regions.

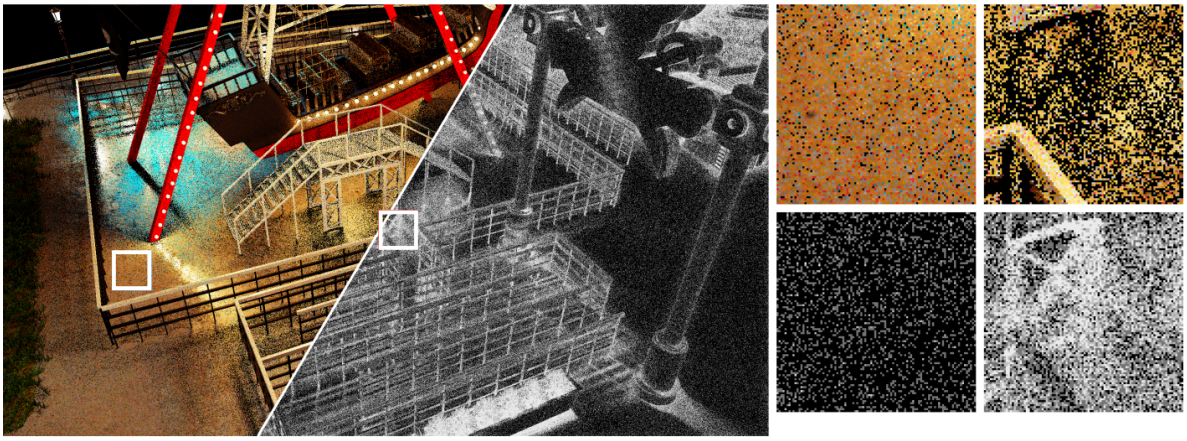


Figure 3.1: Demonstration of duplicate sample prevalence in direct illumination with a four sample reservoir. (Left) Amusement Park scene rendered with ReSTIR for direct illumination. (Right) A heatmap visualising duplicate samples. Black means no duplicates; white means all four light samples are identical. [42]

This problem arises in two scenarios:

- **Initial candidate generation:** The application of WRS in ReSTIR as detailed in Algorithm 3 processes each sample N times, where N is the number of desired output samples. As such, a single sample can potentially pass the check on line 7 several times, resulting in its duplication.
- **Spatiotemporal resampling:** This is caused by different pixels generating identical candidates and being unfortunate enough to select the same candidates for their final samples.

3.3.2. Dividing Candidates Into Subsets

Selecting duplicate samples from our initial candidates would prove detrimental to our approach, as we rely on significantly higher sample counts than are used in most (real-time) applications of ReSTIR. This is needed in order to compute good estimates for optimal weights.

To address this, we effectively divide our candidates into subsets which are each processed by a single ‘sub-reservoir’. Each ‘sub-reservoir’ represents an independent sample, sum of weights, and confidence weight. This is largely based on the stratified version of the RIS estimator as proposed by [35] (Section 4.5 Stratified Resampled Importance Sampling).

Upon receiving a new candidate, this candidate is processed by the sub-reservoir with the smallest sum of weights. This minimises the maximum variance of the unbiased contribution weight of each reservoir. Further, this ensures that high weight candidates are not forced to compete with one another for the same sub-reservoir, increasing the likelihood that multiple high weight (i.e. potentially high quality) candidates will be selected during the initial sampling step. This leads to the reservoir data structure

and corresponding update function in Algorithm 5. Note that this increases storage costs compared to standard WRS due to the additional sum of weights and confidence weight terms that must be stored for each sub-reservoir.

Algorithm 5 Weighted Reservoir Sampling w/ Subset Division and Confidence Weights

class Reservoir
  $Y \leftarrow \varnothing$ ▷ Set of N output samples
  $w_{sum} \leftarrow 0$ ▷ Sum of weights for each sub-reservoir
  $c \leftarrow 0$ ▷ Confidence weight of each output sample
  function update(x_i, w_i, c_i) ▷ New candidate, its weight, and its confidence value
    $j \leftarrow \arg\min w_{sum}$
    $w_{sum_j} \leftarrow w_{sum_j} + w_i$
    $c_j \leftarrow c_j + c_i$
    if `rand()` $< (w_i / w_{sum_j})$ **then**
      $Y_j \leftarrow x_i$
    end if
  end function
end class

3.4. Estimator and Optimal Weights

3.4.1. Combining Samples

Neighbours and Shift Mappings as Distributions

When combining samples from neighbouring pixels under GRIS, we need to apply a shift mapping to map samples to the domain of integration of the target pixel. Different shift mappings can produce different target samples from the same set of source samples. Thus, each (pixel, shift mapping) pair represents a different sampling distribution for the purpose of sample combination.

To illustrate, we return to the scenario depicted in Figure 2.19. Applying the *reconnection shift* shift mapping results in the target sample depicted in Figure 2.20. We now also assume that all surfaces except those of the white box and the ceiling are glossy, i.e. the white box's sides and the ceiling are the only diffuse surfaces in the scene.

A different shift mapping known as the *hybrid shift* postpones reconnection until two consecutive diffuse vertices are encountered along each path; paths are then connected to each other via the second diffuse vertex [25]. Applying this shift mapping to our scene with this additional assumption results in the sample depicted in Figure 3.2.

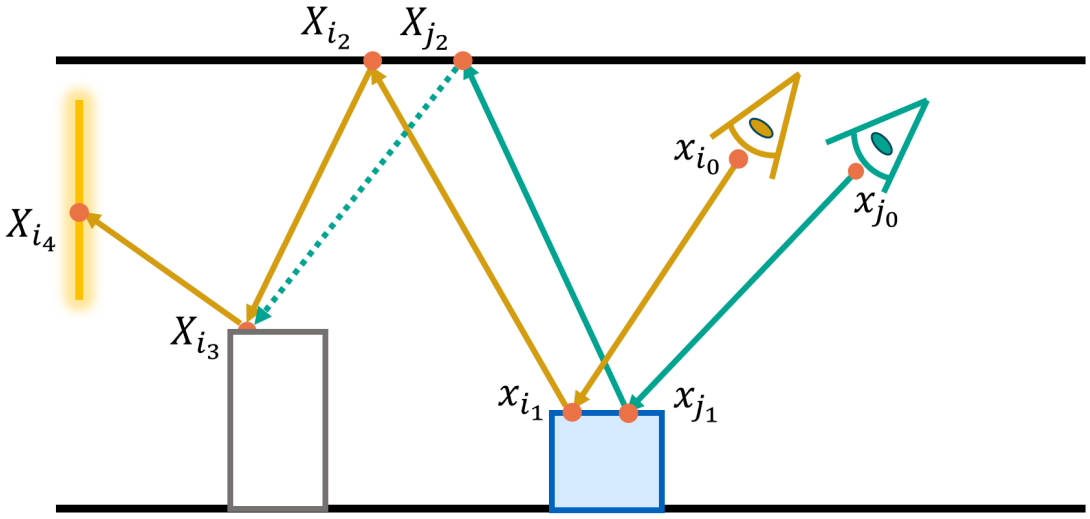


Figure 3.2: Shifting path samples using the *hybrid shift* shift mapping. This particular shift mapping postpones reconnection until two consecutive diffuse vertices are encountered along each path.

R-MIS Estimator

To combine samples generated from multiple pixels, we present the **R-MIS (Resampled Multiple Importance Sampling)** estimator of the form Equation (3.1). N is the total number of (pixel, shift mapping) pairs. n_i corresponds to the number of samples drawn from each (pixel, shift mapping) pair. All other terms correspond to the definitions presented in Sections 2.4.1 and 2.7.2.

$$\langle F \rangle_{RM} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{w_i(Y_{ij}) f(Y_{ij}) W_{Y_{ij}}}{n_i} \tag{3.1}$$

3.4.2. Arbitrary Unbiased Contribution Weights

Estimating optimal weights as outlined in Section 2.4.2 requires evaluating the PDF of distributions from which samples are drawn given arbitrary samples. This cannot be done in a straightforward fashion however, as our samples are generated via RIS and thus have an intractable distribution (which is proportional to the target function) and thus no PDF that we can evaluate.

In line with GRIS/RIS theory, we use the reciprocal of the unbiased contribution weight as an unbiased estimate of this intractable PDF. To compute this weight for arbitrary samples, we emulate replacing the sample associated with a particular unbiased contribution weight with the sample we wish to evaluate this weight for.

We assume that all pixels generate the same number of final samples (and thus have the same number of sub-reservoirs) and generate the same number of initial candidates M. Additionally, we also store the resampling weights of the final samples chosen by the sub-reservoirs w_{final}. Given a sample $Y_{ij} = T(X_{ij})$ produced by shift mapping the jth sample produced by pixel i, we evaluate the reciprocal of the arbitrary unbiased contribution weight at pixel k as[1]

$$\mathcal{W}_k(Y_{ij}) = \begin{cases} \frac{\hat{q}_k(Y_{ij})}{w_{sum_j} - w_{final_j} + w_{Y_{ij}}} & \text{if shift is successful} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

We define the mock[2] resampling weight $w_{Y_{ij}}$ of the shifted sample Y_{ij} as

$$w_{Y_{ij}} = \frac{1}{M} \cdot \hat{q}_k(Y_{ij}) \cdot \frac{1}{p_{ij}(X_{ij})} \cdot |T'(X_{ij})| \tag{3.3}$$

$p_{ij}(X_{ij})$ is the evaluation of the PDF from which X_{ij} was generated *in the source domain*.

3.4.3. Evaluating Estimators

The final form of the proposed R-MIS estimator using optimal weights largely follows the structure presented in Section 2.4.2. A (biased) direct or progressive estimator can be employed. The key difference is that the scaling factor S_{ij} and column vector $\mathbf{W}_{ij}$, which are used to estimate the optimal weights, are now defined as

$$S_{ij} = \frac{1}{\sum_{k=1}^{N} n_k \mathcal{W}_k(Y_{ij})} \quad \mathbf{W}_{ij} = S_{ij}(\mathcal{W}_1(Y_{ij}), ..., \mathcal{W}_N(Y_{ij}))^\intercal \tag{3.4}$$

[1] Though the presented form may seem confusing, it is simply a straightforward reciprocal of the unbiased contribution weight presented in Section 2.5.1 where w_{sum} is altered as described.

[2] The term 'mock' is used to denote the fact that this resampling weight did not originate organically through the RIS resampling procedure.

4

Implementation

4.1. Overview

To assess the practical applicability of the presented technique, we implement it in a custom C++ ray tracer built with Intel's Embree CPU ray tracing framework [40]. We apply this technique to direct illumination. In this context, samples are discrete points on light sources. Note that this does not fully implement the presented theory, as the samples generated by different pixels all lie in the same domain and hence no shift mapping is required when resampling across pixels.

Three distinct ray-tracing 'modes' are provided:

- **R-OMIS**: The estimator detailed in Section 3.4 using optimal weights.
- **R-MIS**: The estimator detailed in Section 3.4 using non-optimal (e.g. generalised balance heuristic) weights.
- **ReSTIR+**: An estimator based on the approach detailed in [3], augmented with our subset division WRS.

4.2. Ray Tracer Specifics

The ray tracer makes use of a Phong material model with diffuse and specular components and a quadratic falloff for the geometry component of the rendering equation. For generating initial light sample candidates, uniform sampling of all lights in the scene is carried out with unshadowed contribution (i.e. Equation (2.21)) as the target function. Lights are represented as distinct entities and can be either points, line segments, quadrilaterals, or circular disks. Line segments, quadrilaterals, and disks are sampled uniformly across their length/surface area.

4.3. Parameters

The parameters that can be adjusted in the implementation can be seen in Figure 4.1. The 'Features' and 'Parameters' blocks contain options which can affect either all modes (Common), or specific modes. The functionality of these parameters is as follows:

- **Neighbour Selection Heuristics**
  - **Same Geometry**: Toggle indicating if neighbours need to be observing the same geometry to be considered similar.
  - **Max depth difference fraction**: Maximum fractional difference between depths, beyond which pixels are considered dissimilar.
  - **Max normal angle difference**: Maximum difference between angles of primary hit normals in degrees, beyond which pixels are considered dissimilar.
- **Features**

  - **Initial Samples - Visibility check**: If toggled, a shadow ray is traced for the chosen sample arising from the initial candidate generation step. The unbiased contribution weight of the sample is set to 0 if the sample is occluded
  - **Save alphas visualisation**: If toggled, the values of the α vectors (see Section 2.4.2) are saved as an image for each color channel and for each pixel.
  - **Use unbiased combination**: If toggled, unbiased combination of reservoirs is performed for the spatial resampling step. Otherwise, biased combination is used. See Appendix A for exact algorithmic details.
  - **Spatial reuse**: Carry out the spatial reuse step.
  - **Spatial reuse - Visibility check**: If toggled and unbiased combination is used, the target function used for spatial reservoir resampling includes a visibility term and thus a shadow ray is traced for each neighbour.
  - **Temporal reuse**: Carry out the temporal reuse step
- **Parameters**
  - **Ray tracing mode**: Select between the three provided 'modes'.
  - **Samples per reservoir**: Number of final samples to store for each reservoir.
  - **Canonical sample count**: Number of initial candidates to generate.
  - **Neighbours to sample**: Number of neighbouring pixels to resample from.
  - **Spatial resample radius**: Pixels in a square centered around the canonical pixel are considered candidates for spatial resampling. The side length of the square is two times this value.
  - **Max iterations**: Number of iterations used for the estimators.
  - **Neighbour selection strategy**
    - **Random**: Neighbours are chosen randomly.
    - **Similar**: Neighbours are selected if they are considered similar according to the pixel similarity heuristics.
    - **Dissimilar**: Neighbours are selected if they are considered *dissimilar* according to the pixel similarity heuristics.
    - **EqualSimilarDissimilar**: An equal number of similar and dissimilar neighbours is chosen.
  - **MIS Weights**: The MIS weights used for the non-optimal R-MIS estimator.
    - **Equal**: Equal weights are assigned to all pixels.
    - **Balance**: Generalised balance heuristic weights are used (see Equation (2.28)). The target function used is unshadowed contribution (i.e. the visibility term of the integrand function is omitted).
  - **Progressive estimator**: If toggled, the progressive variant of the R-OMIS estimator is used. Otherwise, the direct variant is used. See Section 2.4.2.
  - **Progressive update modulo**: This is the U variable (Algorithm 1) which controls the frequency with which the α estimates (and thus the optimal weights estimates) are updated.
  - **Spatial resampling passes**: Number of spatial resampling rounds carried out during the ReSTIR+ spatial resampling step.
  - **Temporal M clamp**: Value of the M*-cap*. When performing temporal resampling with ReSTIR+, the temporal predecessor's confidence weight is capped to a multiple of the canonical pixel's confidence weight. This controls the value of the multiple.

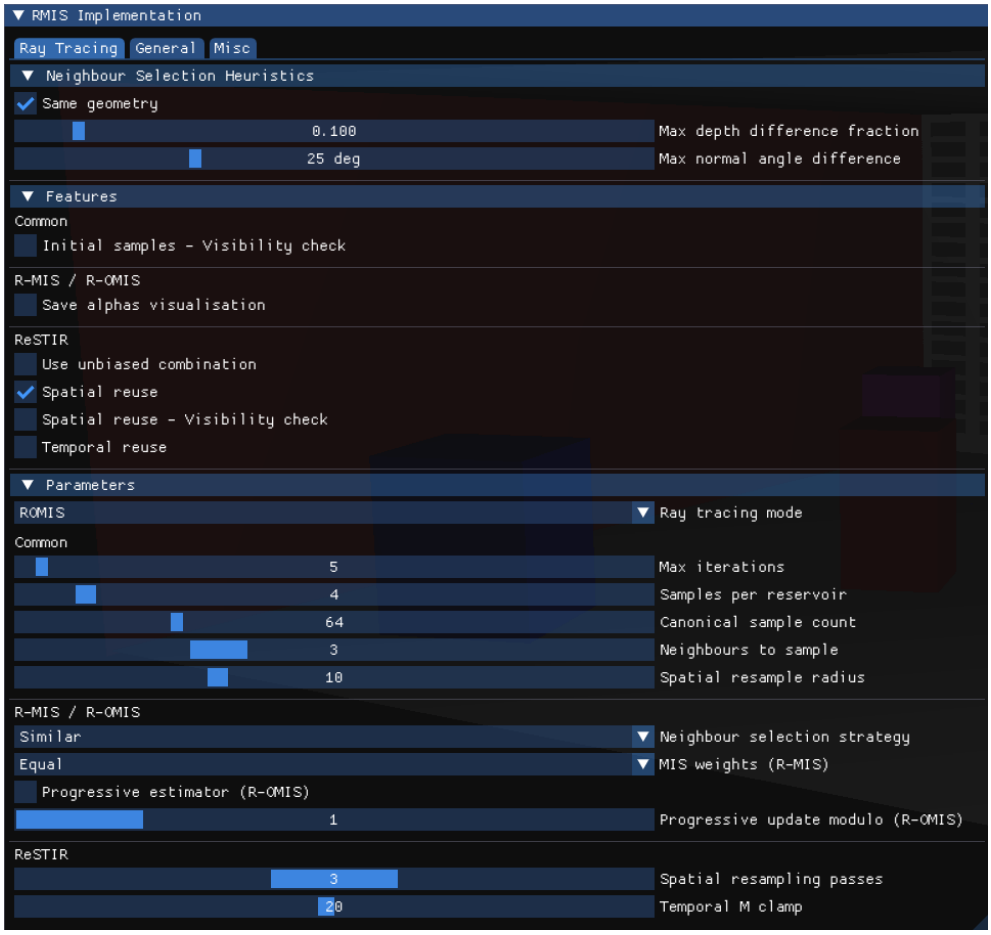


Figure 4.1: Adjustable parameters in the provided implementation.

5

Results and Discussion

5.1. Preliminaries

5.1.1. Testing Environment

All presented results have been rendered with the described renderer using an Intel i7-9750H CPU and 16GiB of RAM at a resolution of 1280×720.

5.1.2. Primary Test Scene

In order to facilitate rapid evaluation and provide a controlled environment that can accentuate aspects of the presented technique, a custom test scene is used for the qualitative results. This test scene is based on the eponymous Cornell Box and is affectionately dubbed the 'Cornell Nightclub'. The scene is effectively a larger Cornell Box with several more boxes in the scene. A render of this scene can be seen in Figure 5.1.

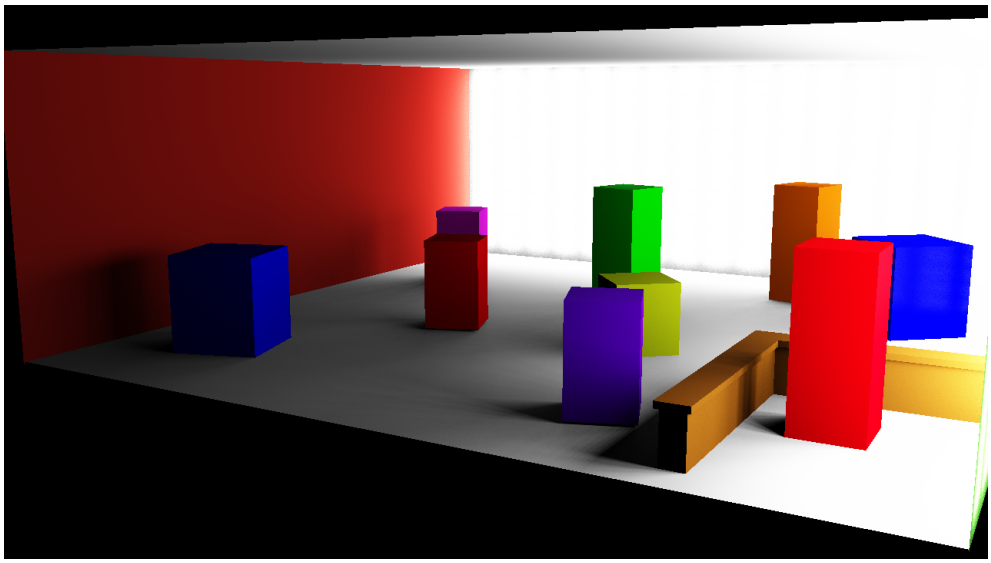
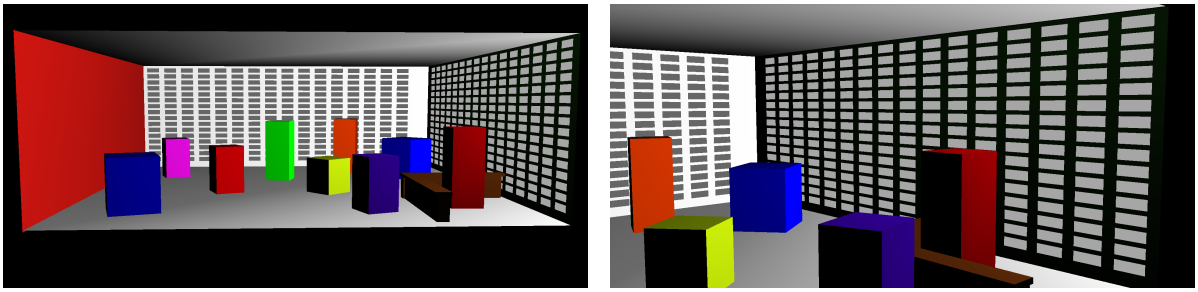


Figure 5.1: A reference render of the Cornell Nightclub test scene.

Where the test scene differs most is in the lighting arrangement as shown in Figure 5.2. The original Cornell Box features a single area light at the top of the box, whereas the Cornell Nightclub features 512 uniformly arranged rectangular area lights. The back wall contains 256 such lights and each one has as color $(0.4, 0.4, 0.4)$; the right wall contains the other 256 and each one has as color $(0.65, 0.65, 0.65)$.



(a) Back wall containing dark grey lights.

(b) Right wall containing light grey lights.

Figure 5.2: Rasterized view of the Cornell Nightclub scene showcasing lighting arrangement.

5.1.3. Additional Test Scenes

To facilitate quantitative analysis under different conditions, three additional scenes are used for producing results. They are adaptations of scenes from Blend Swap, a website where users of the Blender 3D animation and modelling software can share resources. The utilised versions include mild modifications by [2]. These scenes are:

- **Modern Hall**: A hall and staircase illuminated by a planar light at the far end of the hall, several large disc lights on the roof, and smaller planar lights to the right of the staircase. Courtesy of Blend Swap user NewSee2l035.
- **The Breakfast Room**: A dining table illuminated by two overhead circular lamps. Courtesy of Blend Swap user Wig42.
- **The Modern Living Room**: A living room illuminated by a single overhead disc light that is diagonally opposite to the couch. Courtesy of Blend Swap user Wig42.

(a) Modern Hall

(b) The Breakfast Room

(c) The Modern Living Room[1]

Figure 5.3: Reference renders of the scenes used for quantitative analysis.

[1]The back wall of the shelves contains blotchy black artifacts due to geometry overlap with the back wall of the room itself. The placements and sizes of these blotches are consistent and so do not preclude the scene's usability for analysis.

5.1.4. Chosen Parameters

Unless otherwise specified, the relevant parameters used during testing are as follows:

- **Neighbour Selection Heuristics**
  - **Same Geometry**: True
  - **Max depth difference fraction**: 0.1
  - **Max normal angle difference**: 25°
- **Parameters**
  - **Samples per reservoir**: 4
  - **Canonical sample count**: 64
  - **Neighbours to sample**: 3
  - **Spatial resample radius**: 10
  - **Max iterations**: 5
  - **Neighbour selection strategy**: Similar
  - **Progressive estimator**: False (i.e. the direct estimator is used)

5.1.5. Error Metric

The metric used to measure error in this section (relative to a reference render of the scene) is **MAPE (Mean Average Percentage Error)**. This error is computed for an entire image via Equation (5.1), where n denotes the total number of pixels. As the estimated value $\langle I \rangle$ and the ground truth value I both consist of RGB vectors, we compute the MAPE using the L2 norm of the vectors. We denote the L2 norm of vector A as A_2.

$$E = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{I_{i_2} - \langle I \rangle_{i_2}}{I_{i_2}} \right| \tag{5.1}$$

5.1.6. False Colour Insets

Throughout this section, false colour insets will be used to visualise the α vectors produced by the rendering process (see Section 2.4.2). Each pixel associates a value in the range $[-1, 1]$ with itself and each of the neighbours it resamples from. This value correlates with the average weight associated with each pixel from which samples are generated and thus allows us to visualise the sign and magnitude of the computed weights. Lastly, in practice, an integral is estimated for each of the three red, green, and blue colour channels. Thus, each colour channel is associated with a separate α vector and hence a separate visualisation.

Figure 5.4 shows an example of such an inset. The colour bar on the right indicates the colour associated with different values. Positive values are depicted in orange, while negative values are depicted in blue.

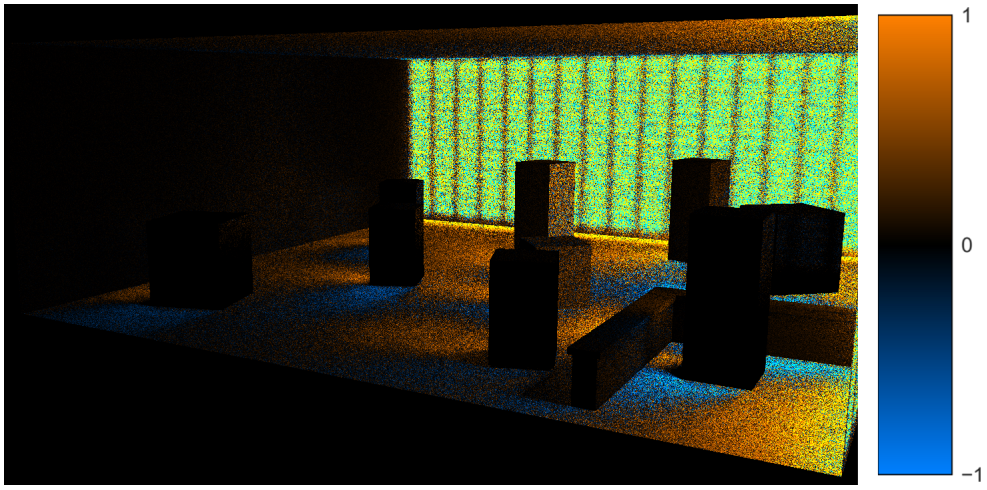


Figure 5.4: Example of a false colour inset for the green channel.

5.2. Similarity Heuristics

Figure 5.5 compares the results of using different strategies for selecting neighbours for resampling. The heuristics specified earlier are used to select either similar neighbours, dissimilar neighbours, or an equal number of both. The last strategy randomly selects neighbours in the user-specified neighbourhood. Most notably, preferring dissimilar pixels creates visible bands around geometric discontinuities, likely as a result of the mismatch between the shapes of target functions between dissimilar pixels. The 'Similar' strategy shows the smallest amount of banding, followed by random selection, the 'Equal' strategy, and then the 'Dissimilar' strategy. This demonstrates the importance of selecting similar neighbours for our approach, as they are more likely to provide useful samples that do not add variance to the estimator.

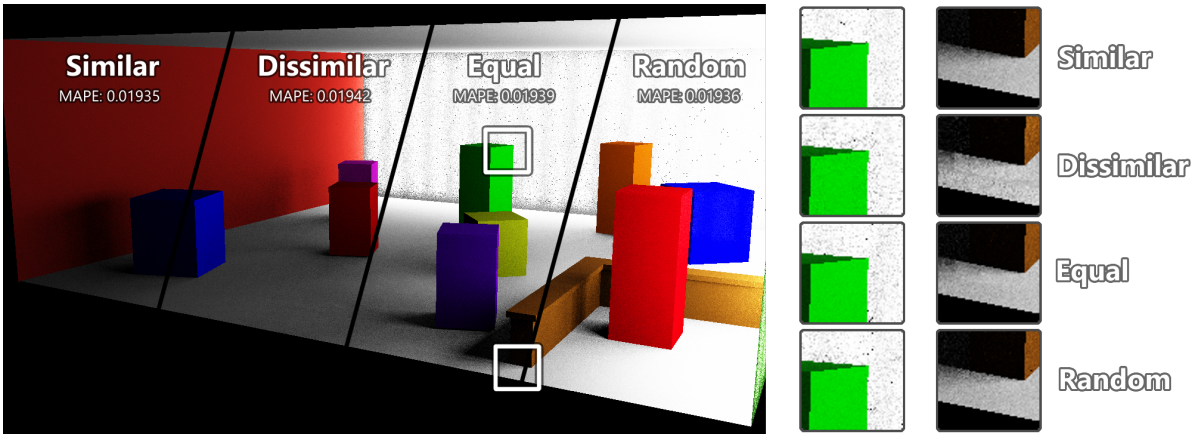
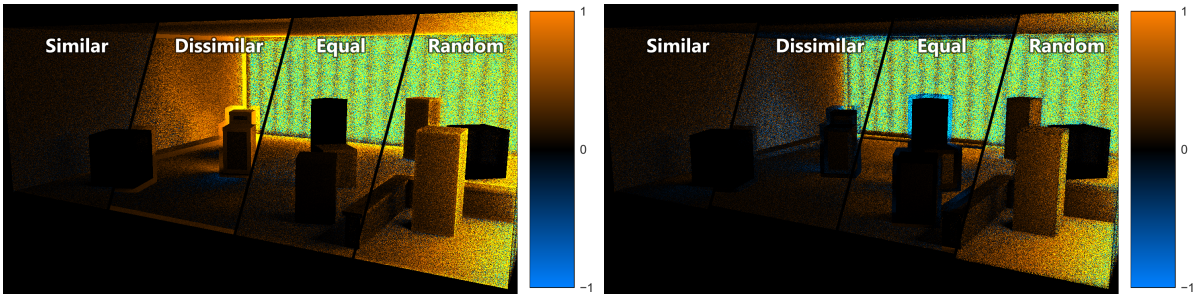


Figure 5.5: Comparison of similarity heuristic selection techniques.

The aforementioned bands are reflected in the generated α vectors and thus the subsequent weights, as can be seen in Figure 5.6, which shows these values for the canonical pixel and one of its neighbours. The prevalence of these bands can be further seen in Figure 5.7, which showcases the regions inspected in the zoomed in snippets from Figure 5.5. Optimal weight estimates appear to discourage samples from dissimilar pixels by assigning them intensely negative weights.



(a) Canonical **(b)** Neighbour

Figure 5.6: Comparison of α vectors generated under different selection techniques for the red channel.

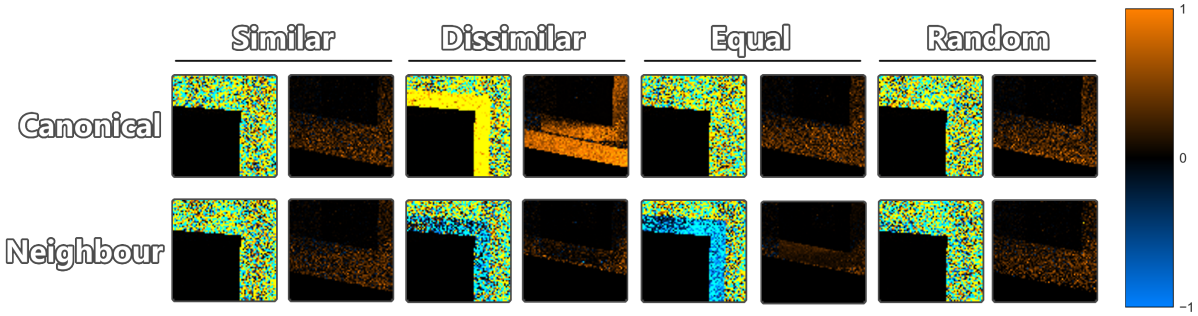


Figure 5.7: Zoomed in views of the α vectors generated under different selection techniques for the red channel.

5.3. Canonical Candidate Count

Figure 5.8 shows the effect of increasing the number of candidates used for the initial candidate generation step. In addition to yielding higher quality samples, increasing the candidate count reduces the variance of the UCW [25]. This results in arbitrary UCWs with lower variance, which reduces the variance of the $\langle \mathbf{A} \rangle$ and $\langle \mathbf{b} \rangle$ estimators, yielding estimated optimal weights with lower variance.

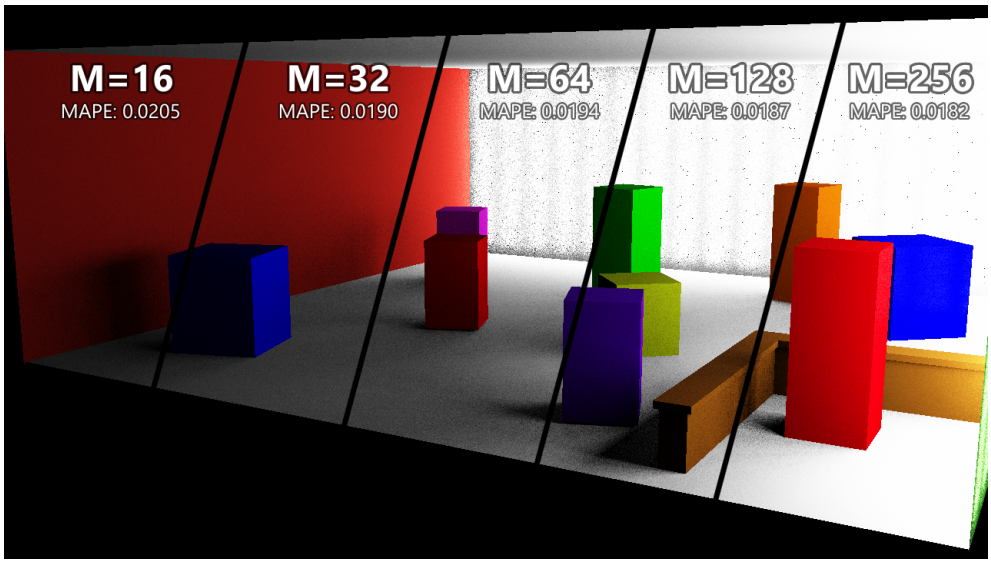


Figure 5.8: Comparison of varying numbers of initial light candidates.

Figure 5.9 showcases a reduction in the variance of the α vector estimates and thus the aforementioned subsequent reduction in the variance of the optimal weight estimates. The false colour insets become generally less grainy and regions with smoothly varying weights of a particular sign (positive or negative) begin to manifest. This is not the case across the entire image however, as evidenced by the back-left wall and right side of the box in the first zoomed in view (of one of the boxes in the middle of the nightclub).

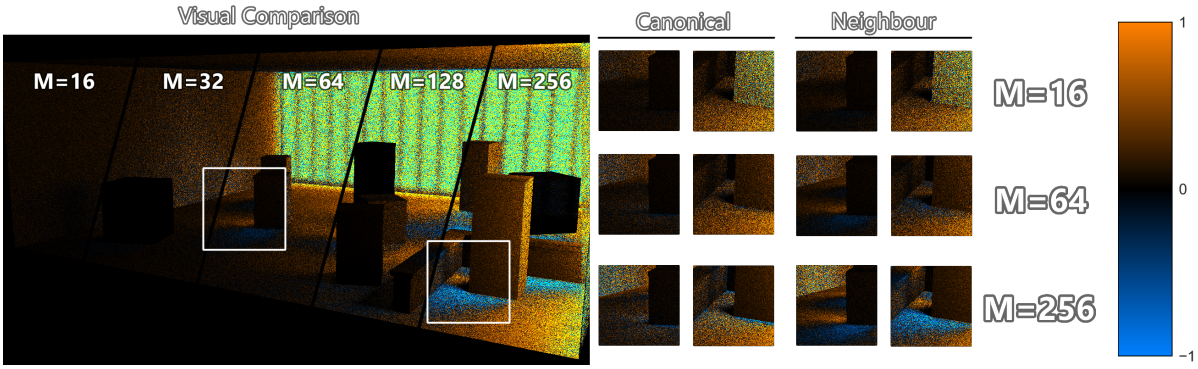


Figure 5.9: Effect of increasing the number of initial light candidates on α vector estimates for the red colour channel of the canonical pixel. The zoomed in views highlight two exemplary areas for both the canonical pixel and one of its resampling neighbours.

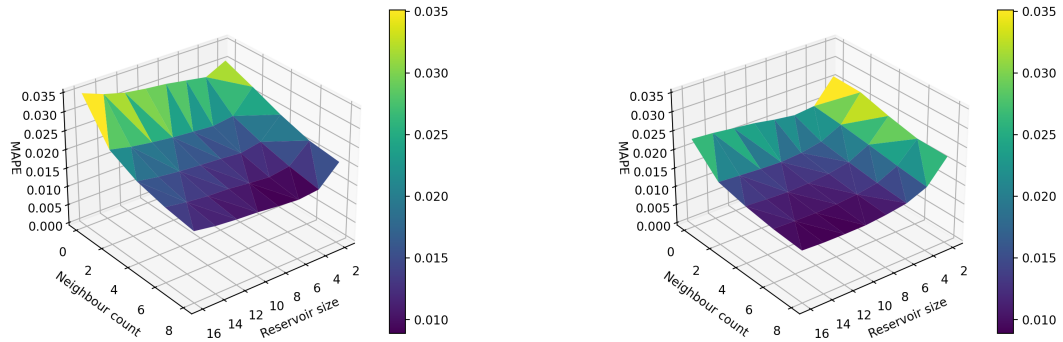
5.4. Reservoir Size and Neighbour Count

In addition to their typical role in ReSTIR-style approaches, the number of samples stored per reservoir and the number of neighbours being resampled from play an additional role in our approach. Increasing the number of samples from each neighbour allows us to derive more accurate estimates for the α vectors and subsequent optimal weights, but increases the size of the linear system that must be solved. Additional neighbours provide more (potentially high quality) samples, but also increase the size of the linear system.

Figures 5.10 to 5.13 show how MAPE varies with differing reservoir sizes and resampling neighbour counts for the tested scenes. We show this variation with different numbers of candidates for the initial candidate generation step. MAPE generally decreases with increasing reservoir sizes, provided that the number of candidates grows in relative proportion to reservoir size. With a low number of candidates, we do not acquire samples that are well distributed according to the target function and (more importantly) our UCWs are high variance, leading to high variance in our optimal weight estimates. This leads to the counter-intuitive effect of larger reservoirs producing worse results with a low number of initial candidates. Given sufficient initial candidates, increasing reservoir size produces better results.

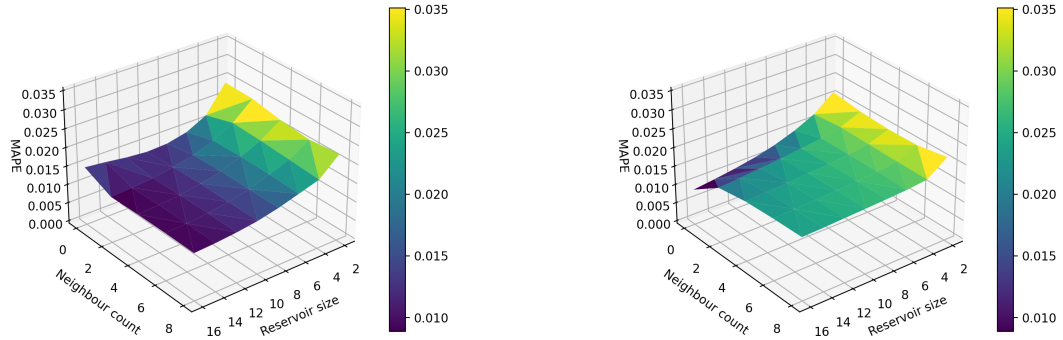
Increasing the number of neighbours reduces MAPE in instances where reservoir size is not large and the number of initial candidates is relatively small, allowing for pixels to make up for their lack of well-distributed samples by borrowing from neighbours. As canonical candidate count increases, the benefit of borrowing from neighbours gradually decreases, eventually yielding worse results. This is likely due to the simplicity of most of the tested scenes resulting in pixels being able to produce sufficiently high quality samples given enough candidates. Samples from neighbours are thus not as high quality as those produced by the pixel itself in those instances.

These observations suggest that a balancing act exists between reservoir size and neighbour count in relation to candidate count. In scenarios where it is expensive to produce several initial candidates, increasing neighbour counts both improves final quality and amortizes sample generation costs. Geometric complexity and the abundance of smoothly varying geometric features appears to dictate this balancing act. This is demonstrated by the results from ‘The Modern Living Room’ in Figure 5.13, which violate the trend seen in the other scenes. Additional neighbours and larger reservoirs lead to universally better results, likely owing to the greater geometric variety of the scene. This result is counter-intuitive, as greater geometric variety should lead to worse neighbour samples due to the resulting dissimilarity in target functions.



(a) 16 canonical candidates

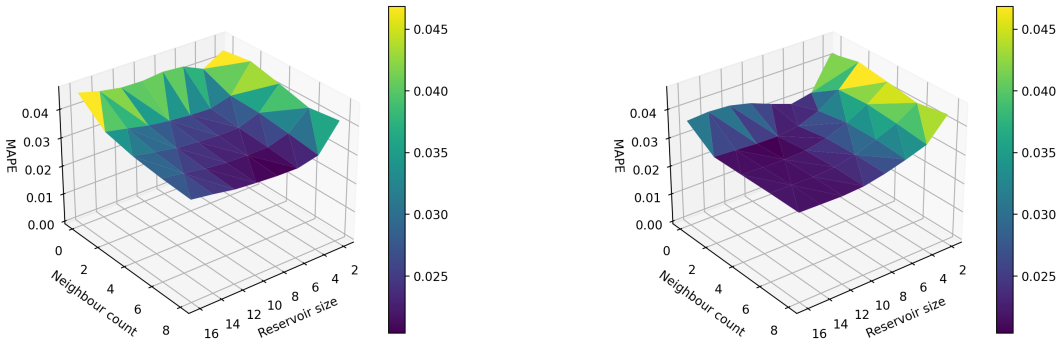
(b) 32 canonical candidates



(c) 64 canonical candidates

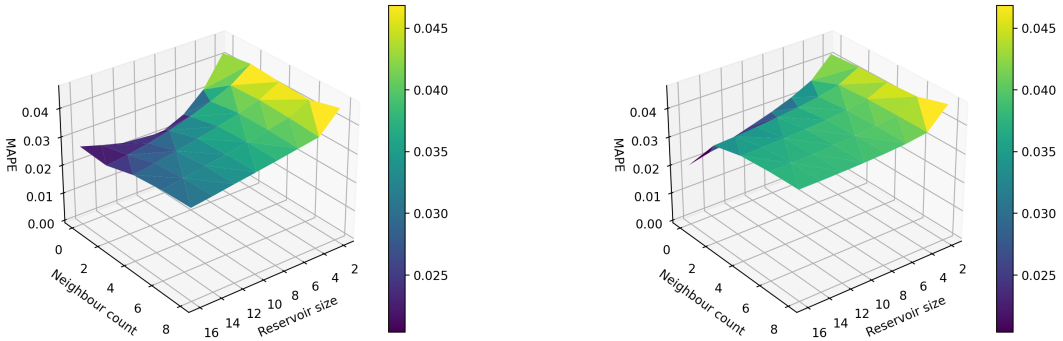
(d) 256 canonical candidates

Figure 5.10: Cornell Nightclub



(a) 16 canonical candidates

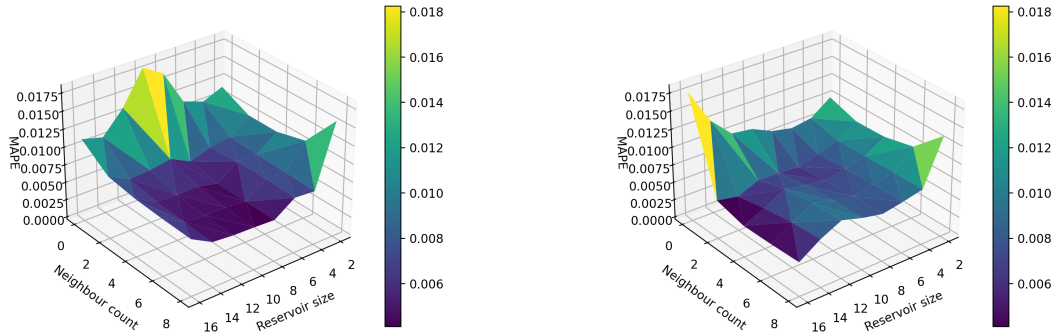
(b) 32 canonical candidates



(c) 64 canonical candidates

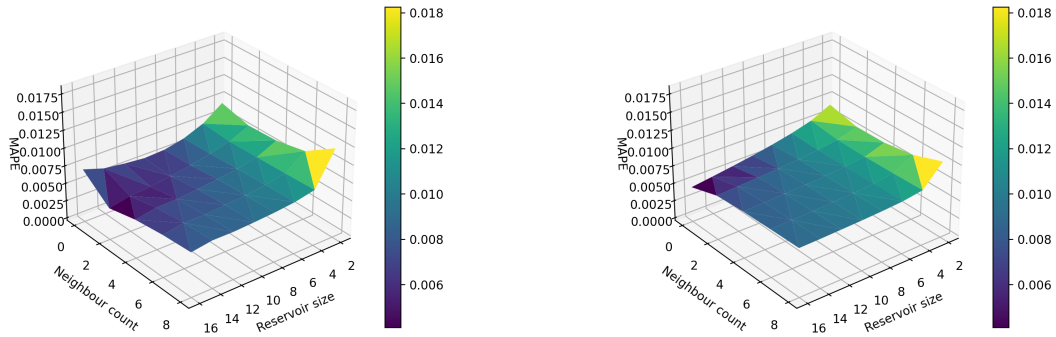
(d) 256 canonical candidates

Figure 5.11: Modern Hall



(a) 16 canonical candidates

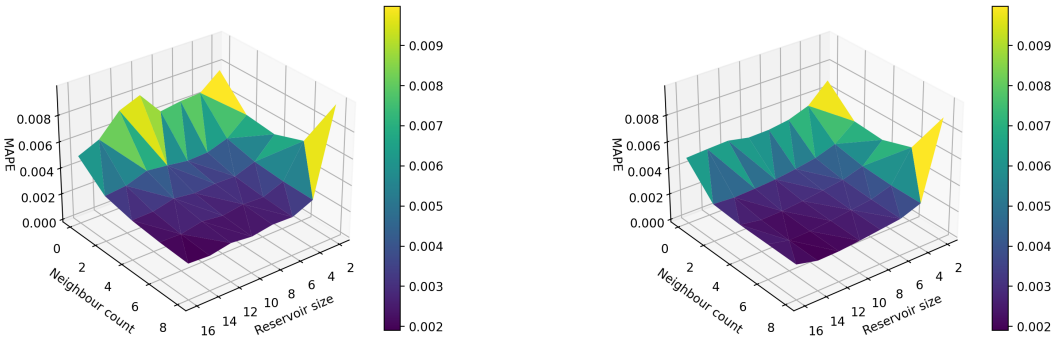
(b) 32 canonical candidates



(c) 64 canonical candidates

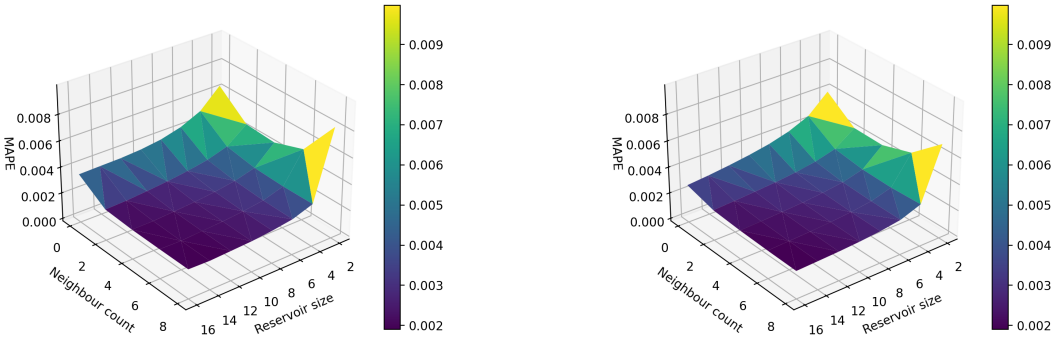
(d) 256 canonical candidates

Figure 5.12: The Breakfast Room



(a) 16 canonical candidates

(b) 32 canonical candidates



(c) 64 canonical candidates

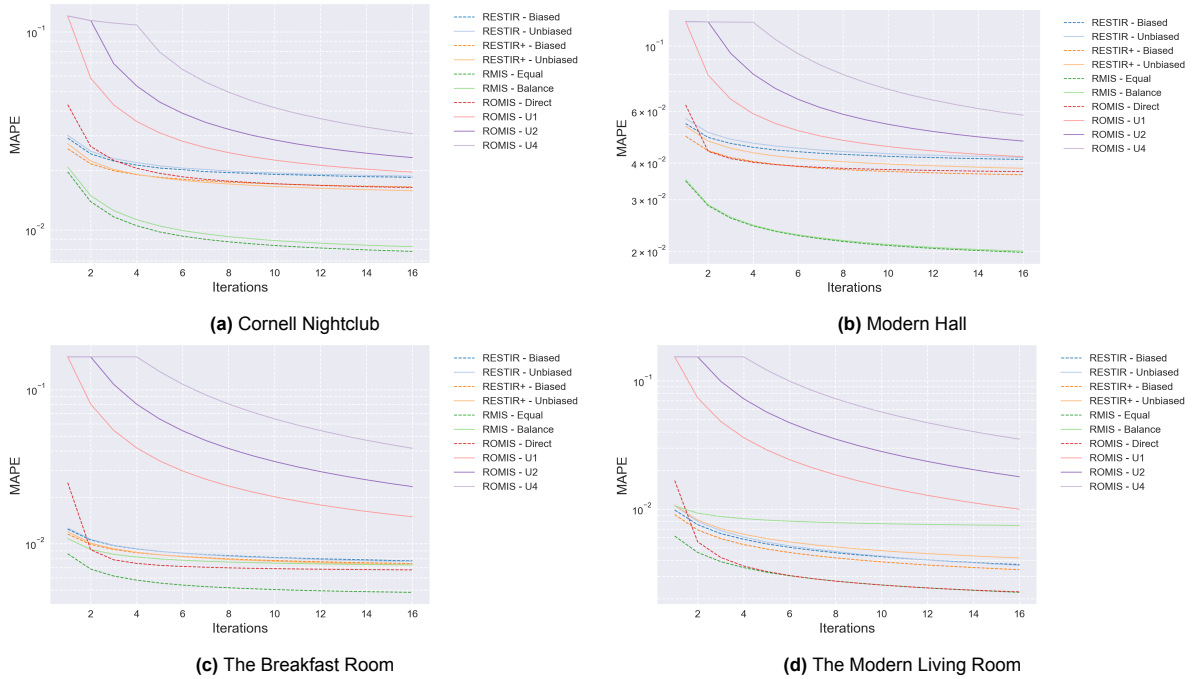
(d) 256 canonical candidates

Figure 5.13: The Modern Living Room

5.5. Convergence Behaviour

Figure 5.14 shows how MAPE scales with increasing iterations for all tested scenes. The tested configurations are as follows:

- **ReSTIR**: A baseline implementation of the original ReSTIR methods presented in [3]. We carry out 3 rounds of spatial resampling, do not conduct temporal resampling, and average the results of several iterations as in the offline version of the estimator from [25]. We do not conduct a visibility check for spatial resampling (i.e. the target function used for spatial resampling does not include a visibility term). We test versions with both biased and unbiased reservoir combinations (algorithms 4 and 6 from [3]).
- **ReSTIR+**: Our variation of ReSTIR incorporating WRS with subset division. Similarly to ReSTIR, we test versions with both biased and unbiased reservoir combinations (see Appendix A for details on the reservoir combination algorithms).
- **R-MIS**: The presented R-MIS estimator with equal and balance heuristic MIS weights.
- **R-OMIS**: The presented R-OMIS estimator. We test both the Direct and Progressive versions. For the progressive estimator, values of 1, 2, and 4 are used for the U parameter controlling the frequency with which the α vector estimates are updated.



(a) Cornell Nightclub

(b) Modern Hall

(c) The Breakfast Room

(d) The Modern Living Room

Figure 5.14: Decrease in error (MAPE) over iterations for the tested scenes.

The first observation of interest is that ReSTIR+ generally outperforms ReSTIR in both of their biased and unbiased variations, showcasing the benefits afforded by lower sample duplication probabilities from subset division. The only exception is in 'The Modern Living Room' where unbiased baseline ReSTIR outperforms unbiased ReSTIR+. This is likely due to the resulting UCWs being of a higher variance with ReSTIR+, as fewer samples contribute to the UCW of each final sample as a result of storing separate w_{sum} terms for each sub-reservoir. This variance is further exacerbated by the unbiased combination as it effectively 'invalidates' the contribution of some samples to the UCW of the final samples compared to the biased combination.

Furthermore, R-MIS generally outperforms all ReSTIR variants. This comes as no surprise, as each R-MIS iteration shades significantly more samples compared to ReSTIR. This increases quality at the cost of additional shadow rays in our direct illumination scenario. Strangely, using equal weights yields better results than using generalised balance heuristic weights. This might be due to the generalised

balance heuristic weights exacerbating error due to their omission of the visibility term. Experimentation with additional heuristics such as pairwise weights [1] and ratio weights [27] may yield better results. Additionally, in applications where the integrand function is used as the target function (such as path tracing), these results may differ drastically.

Lastly, the Progressive versions of R-OMIS are the worst performing by a significant margin. The Direct version performs comparably to the ReSTIR variants in scenes having low geometric complexity (and thus greater similarity between target PDFs of neighbouring pixels). In 'The Breakfast Room', it outperforms the ReSTIR variants as well as R-MIS with balance heuristic weights. In 'The Modern Living Room', it closely matches R-MIS with equal weights as of the fifth iteration. This suggests that optimal weights are largely not beneficial unless the scene has sufficient geometric and lighting complexity which would lead to the similarity heuristics not being able to identify good neighbours to resample from.

5.6. Runtime Analysis

Figure 5.15 showcases runtimes for a single iteration of the configurations tested in Section 5.5, broken down by their respective steps. For 'R-OMIS Progressive', we include the time needed to update the α vector estimates. Configurations which share a step and execute it in the same manner are grouped together. The significance of each step is as follows:

- **Primary Rays**: Ray-tracing step which determines the primary hitpoint and its properties.
- **Resampling Indices**: Determination of the similarity of neighbours in the resampling neighbourhood as described in Section 3.2.
- **Candidate Generation**: Generation of samples from an initial set of candidates. This is the first step of the process described in Section 2.6.3.
- **Spatial Resampling**: Resampling reservoirs from neighbour pixels. This is the third step of the process described in Section 2.6.3.
- **Shading**: For ReSTIR-derived approaches, this simply denotes shading the final samples that have survived initial candidate generation and spatial resampling. For R-MIS this denotes computing MIS weights for and shading all gathered samples. For R-OMIS this denotes the work defined per iteration for the estimators from Algorithms 1 and 2.
- **Iteration Combination**: For 'R-OMIS Direct', this denotes solving for the α vectors and computing their sum as described in lines 14 and 15 of Algorithm 2. For all other configurations, this denotes simply averaging the output of the various iterations.
- **Total**: Total execution time for the configuration. Note that summing the other columns will not produce the exact number presented here due to the grouping of steps shared by different configurations.
- **Total - Iteration Work**: Total execution time for the steps carried out each iteration (i.e. total execution time minus the steps carried out before or after the estimators' iterations).

We observe that subset division WRS leads to a faster candidate generation step as evidenced by baseline ReSTIR being slower in this respect compared to the other configurations. This is likely due to not having to generate a random number for each sub-reservoir, as is the case with baseline WRS. The overhead of determining the sub-reservoir with the smallest w_{sum} does not appear to be significant. However, spatial resampling with subset division WRS is significantly more expensive.

The R-MIS estimators appear to be generally faster than the ReSTIR-derived estimators. However, this is a highly deceptive result. The scenes used for our testing are relatively geometrically simple, with triangle counts ranging from 83 for the 'Cornell Nightclub' to 134,701 for 'The Breakfast Room'. Consequently, the cost of tracing rays (and thus the visibility term of the integrand function) is quite low relative to the other terms of the integrand. This is evident from the runtimes for the 'Primary Rays' step across all scenes. In larger scenes with more complex geometry, tracing rays will be significantly more expensive.

More complex MIS weighting schemes are predictably more expensive, as evidenced by the R-OMIS estimators and ‘R-MIS - Balance’ being significantly more costly than ‘R-MIS - Equal’. This further motivates the benefits to be gained by experimenting with alternative MIS weighting schemes.

	Primary Rays	Resampling Indices	**Candidate Generation**	**Spatial Resampling**	**Shading**	Iteration Combination	Total	Total - Iteration Work
ReSTIR - Biased	79	N/A	**3267**	**2970**	**122**	6	6532	6447
ReSTIR - Unbiased				**4094**			7818	7733
ReSTIR+ - Biased			**2724**	**3819**			7553	7468
ReSTIR+ - Unbiased				**5090**			8835	8750
R-MIS - Equal		1559		**N/A**	**744**		5393	3749
R-MIS - Balance					**2192**		6817	5173
R-OMIS - Progressive					**3580**		9103	7459
R-OMIS - Direct					**2786**	984	9240	6618

(a) Cornell Nightclub

	Primary Rays	Resampling Indices	**Candidate Generation**	**Spatial Resampling**	**Shading**	Iteration Combination	Total	Total - Iteration Work
ReSTIR - Biased	91	N/A	**3963**	**3326**	**151**	7	7721	7623
ReSTIR - Unbiased				**4831**			9240	9142
ReSTIR+ - Biased			**3609**	**4391**			8897	8799
ReSTIR+ - Unbiased				**5722**			10114	10016
R-MIS - Equal		1565		**N/A**	**877**		6439	4776
R-MIS - Balance					**2701**		8443	6780
R-OMIS - Progressive					**4236**		10994	9331
R-OMIS - Direct					**3263**	1038	10637	7943

(b) Modern Hall

	Primary Rays	Resampling Indices	**Candidate Generation**	**Spatial Resampling**	**Shading**	Iteration Combination	Total	Total - Iteration Work
ReSTIR - Biased	116	N/A	**4022**	**3181**	**149**	6	7736	7614
ReSTIR - Unbiased				**4491**			8916	8794
ReSTIR+ - Biased			**3559**	**4413**			8927	8805
ReSTIR+ - Unbiased				**5390**			10081	9959
R-MIS - Equal		1616		**N/A**	**918**		6611	4873
R-MIS - Balance					**2579**		8260	6522
R-OMIS - Progressive					**5155**		11654	9916
R-OMIS - Direct					**3086**	2117	11630	7781

(c) The Breakfast Room

	Primary Rays	Resampling Indices	**Candidate Generation**	**Spatial Resampling**	**Shading**	Iteration Combination	Total	Total - Iteration Work
ReSTIR - Biased	92	N/A	**4185**	**3181**	**162**	8	7875	7775
ReSTIR - Unbiased				**4491**			9264	9164
ReSTIR+ - Biased			**3846**	**4413**			9578	9478
ReSTIR+ - Unbiased				**5390**			10595	10495
R-MIS - Equal		1536		**N/A**	**926**		6934	5298
R-MIS - Balance					**2829**		8861	7225
R-OMIS - Progressive					**6354**		13008	11372
R-OMIS - Direct					**3348**	3175	13288	8485

(d) The Modern Living Room

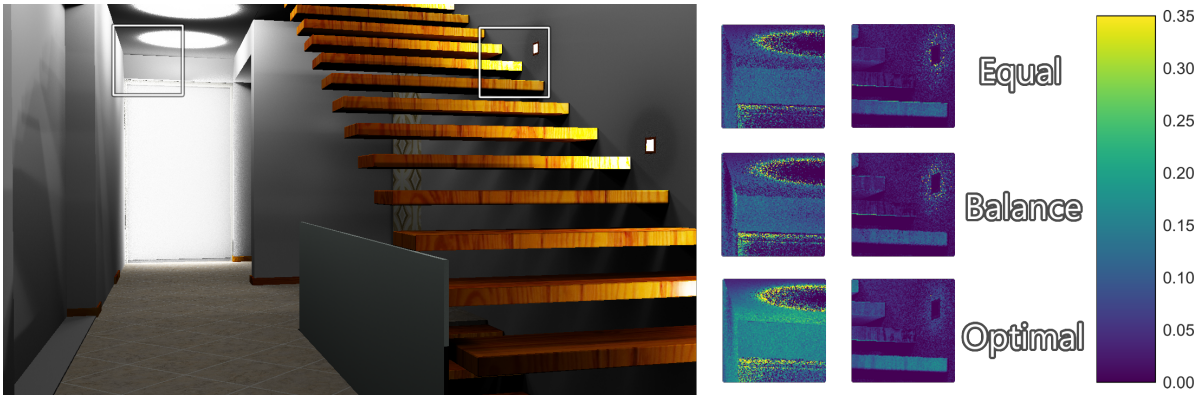
Figure 5.15: Runtimes for the presented estimators for a single iteration, broken down by their respective steps, for each scene. The columns in bold are the steps carried out each iteration; other columns denote steps that are carried out only once per evaluation of the estimator or aggregate times. All times are in milliseconds.

5.7. R-MIS Weights

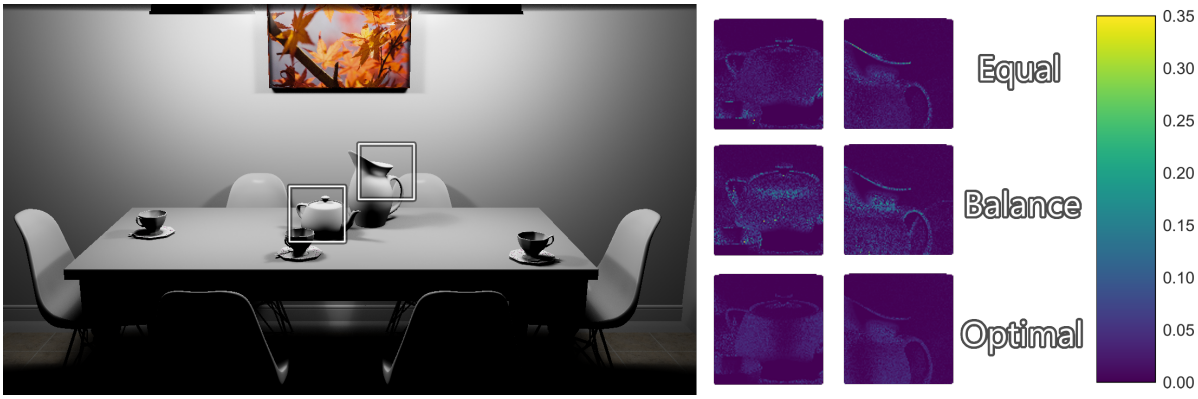
Figure 5.16 demonstrates the effects of different weighting schemes for the R-MIS estimator. We compare to equal weights as a baseline. Balance heuristic weights generally produce worse results, with the exception of regions with extreme geometric dissimilarities between pixels, such as around the rim of the jug in ‘The Breakfast Room’. These extreme cases take advantage of the balance heuristic weights’ ability to combat dissimilarities between pixels’ target functions and discourage obviously suboptimal samples, but such cases are sparse in the tested scenes.

Optimal weights fare the best with geometric dissimilarities, particularly when they vary smoothly such as along the jug, teapot, and teacups in ‘The Breakfast Room’. However, optimal weights do not fare as well compared to equal weights when neighbours are similar in geometric properties as well as relative lighting distributions, such as on the stairs of the staircase in ‘Modern Hall’. Additionally, regions exposed to intense light, such as the back corner in ‘Modern Hall’, pose a significant challenge to optimal weights, which are outperformed by equal and balance heuristic weights.

This suggests that optimal weights can offer greater flexibility in scenes with greater geometric variety and more intricate lighting arrangements, as they provide a mechanism for learning the usefulness of samples generated from different neighbours. This is especially the case if the similarity heuristics are not capable of fully capturing these differences. If a scene does not require such flexibility however, using optimal weights is (ironically) suboptimal.



(a) Modern Hall



(b) The Breakfast Room

Figure 5.16: MAPE heatmaps showcasing error on a per-pixel basis for the R-MIS estimator with equal weights, balance heuristic weights, and the Direct variant of the R-OMIS estimator, labelled 'Optimal'.

5.8. WRS With Subset Division

Figure 5.17 demonstrates the effects of WRS with subset division in isolation by comparing the ReSTIR+ estimator to baseline ReSTIR. ReSTIR+ appears to be more capable of finding good samples in challenging conditions, such as partially shadowed regions and regions of smooth geometric variety. This can be seen in the zoomed in view of the jug's shadow and handle, where ReSTIR+ demonstrates less error compared to baseline ReSTIR for both the biased and unbiased variants.

However, ReSTIR+ is more prone to bright artifacts, commonly referred to as 'fireflies'. These are particularly evident around geometric discontinuities where neighbours are likely to have differently shaped target functions and manifest as isolated spots of high error in the heatmap. This can be seen in the zoomed in view of the rim of the chair and the edge of the table, particularly for the unbiased variants. These results are likely due to variance in the UCW estimates.

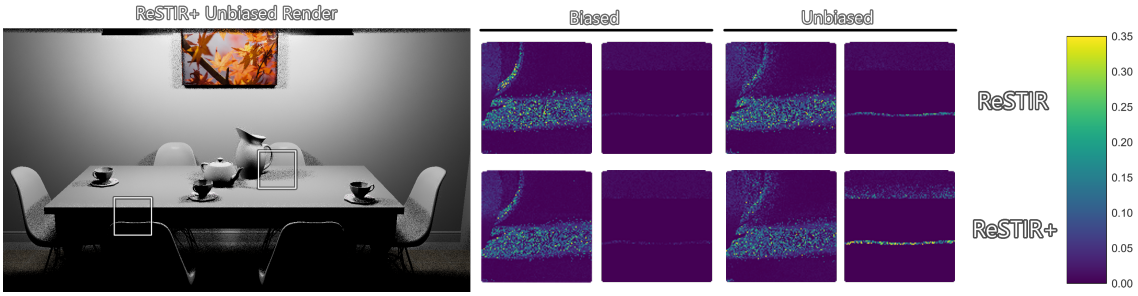


Figure 5.17: Render of 'The Breakfast Room' produced by the unbiased variant of ReSTIR+ alongside MAPE heatmaps showcasing error on a per-pixel basis for the biased and unbiased variants of ReSTIR and ReSTIR+ (detailed in Section 5.5). Note that the corresponding renders use a single iteration of each estimator.

6

Conclusion

We have presented a framework for combining samples from multiple distributions (and generated in potentially different domains) via RIS in an MIS-style estimator. We have extended the optimal MIS weight scheme to this estimator by defining a method for constructing arbitrary unbiased contribution weights for samples not generated via the original RIS procedure which yielded the base unbiased contribution weight. We have shown the importance of selecting neighbours with similarly shaped target functions for the ideal function of the presented estimator. Lastly, we demonstrated a method for extending WRS to handle generating multiple samples with no risk of duplicate samples.

6.1. Future Work

6.1.1. Ablation Study With Subset Division WRS

Subset division WRS appears to be a generally effective augmentation to WRS as evidenced by our ReSTIR+ implementation producing better results compared to baseline ReSTIR. However, it is associated with increased storage costs and increased runtime costs for spatial resampling (in our implementation).

A thorough investigation of the behaviour of subset division WRS would enrich the current scope of ReSTIR literature. Its costs vary differently compared to baseline WRS, relative to parameters such as the number of final samples, the number of initial candidates, and the extent of spatiotemporal resampling. Further, it may help with alleviating the correlation artifacts common in ReSTIR-based approaches. Lastly, other heuristics for selecting the sub-reservoir to be used for processing input samples may prove superior.

6.1.2. Accumulating Unbiased Contribution Weight Estimates

In typical resampling usage scenarios, unbiased contribution weights are computed based on a single round of candidate generation. As the presented approach is tailored towards offline rendering and so involves generating candidates based on the same target function across iterations, it may be possible to accumulate unbiased contribution weight estimates across iterations. This would greatly reduce variance as a result of these estimates. Initial experimentation did not prove successful, suggesting that the involved maths is not straightforward.

6.1.3. Neighbour Selection

Selecting neighbours with target functions whose shapes are similar is crucial for the success of the presented method. While the presented similarity heuristics produce satisfactory results, additional heuristics such as similarity of BRDFs may produce better results.

Further, more advanced techniques that attempt to directly approximate the distribution of lighting around particular points can be used to derive distributions that can be compared for similarity. A pre-processing step involving approaches similar to path guiding [39] or estimation of PDFs as von Mises–Fisher distributions [36] can be used to inform the choice of resampling neighbours.

6.1.4. Application to Scenarios With Non-Trivial Spatial Resampling

The presented implementation applies the proposed method in the context of direct lighting using target functions that omit the visibility term. As such, evaluating the actual integrand function involves tracing an additional visibility ray after resampling has taken place.

In scenarios where the spatial resampling step itself produces a value for the integrand function, such as global illumination [26] and arbitrary light-carrying paths [25, 24], the presented method should be significantly more efficient. In such scenarios, a spatially resampled sample is usually significantly cheaper than a canonical sample. Further, such scenarios usually involve the usage of shift mappings to reuse samples across domains and would constitute a full implementation of the presented theory.

References

[1] Benedikt Bitterli. “Correlations and reuse for fast and accurate physically based light transport”. PhD thesis. Dartmouth College, 2021.

[2] Benedikt Bitterli. *Rendering resources*. https://benedikt-bitterli.me/resources/. 2016.

[3] Benedikt Bitterli et al. “Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting”. In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 148–1.

[4] Jiayin Cao. *Monte Carlo Integral with Multiple Importance Sampling*. Aug. 2015. url: `https://agraphicsguynotes.com/posts/monte_carlo_integral_with_multiple_importance_sampling/`.

[5] Min-Te Chao. “A general purpose unequal probability sampling plan”. In: *Biometrika* 69.3 (1982), pp. 653–656.

[6] Per H Christensen, Wojciech Jarosz, et al. “The path to path-traced movies”. In: *Foundations and Trends® in Computer Graphics and Vision* 10.2 (2016), pp. 103–175.

[7] Ege Ciklabakkal et al. “Single-pass stratified importance resampling”. In: *Computer Graphics Forum*. Vol. 41. 4. Wiley Online Library. 2022, pp. 41–49.

[8] Carsten Dachsbacher et al. “Implicit visibility and antiradiance for interactive global illumination”. In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), 61–es.

[9] Addis Dittebrandt et al. “Markov Chain Mixture Models for Real-Time Direct Illumination”. In: *Computer Graphics Forum*. Vol. 42. 4. Wiley Online Library. 2023, e14881.

[10] Pavlos S Efraimidis. “Weighted random sampling over data streams”. In: *Algorithms, Probability, Networks, and Games: Scientific Papers and Essays Dedicated to Paul G. Spirakis on the Occasion of His 60th Birthday* (2015), pp. 183–195.

[11] Elmar Eisemann and Frédo Durand. “Flash photography enhancement via intrinsic relighting”. In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 673–678.

[12] Luca Fascione et al. “Path tracing in production”. In: *ACM SIGGRAPH 2018 Courses*. 2018, pp. 1–79.

[13] Iliyan Georgiev and Philipp Slusallek. “Simple and Robust Iterative Importance Sampling of Virtual Point Lights.” In: *Eurographics (Short Papers)*. 2010, pp. 57–60.

[14] Iliyan Georgiev et al. “Light transport simulation with vertex connection and merging.” In: *ACM Trans. Graph.* 31.6 (2012), pp. 192–1.

[15] Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. “Multiplexed metropolis light transport”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–10.

[16] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. “A path space extension for robust light transport simulation”. In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), pp. 1–10.

[17] Eric Haines and Tomas Akenine-Möller, eds. *Ray Tracing Gems*. `http://raytracinggems.com`. Apress, 2019.

[18] Eric Heitz et al. “A low-discrepancy sampler that distributes Monte Carlo errors as a blue noise in screen space”. In: *ACM SIGGRAPH 2019 Talks*. 2019, pp. 1–2.

[19] Binh-Son Hua et al. “A Survey on Gradient-Domain Rendering”. In: *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 455–472.

[20] Bernhard Kerbl and Adam Celarek. *Materials I*.

[21] Bernhard Kerbl and Adam Celarek. *Monte Carlo Integration I*.

[22] Ivo Kondapaneni et al. “Optimal multiple importance sampling”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–14.

[23] Jaroslav Křivánek et al. “Unifying points, beams, and paths in volumetric light transport simulation”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–13.

[24] Daqi Lin, Chris Wyman, and Cem Yuksel. “Fast volume rendering with spatiotemporal reservoir resampling”. In: *ACM Transactions on Graphics* 40.6 (Dec. 2021), pp. 1–18. doi: 10.1145/3478513.3480499.

[25] Daqi Lin et al. “Generalized resampled importance sampling: foundations of ReSTIR”. In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–23.

[26] Yaobin Ouyang et al. “ReSTIR GI: Path resampling for real-time path tracing”. In: *Computer Graphics Forum*. Vol. 40. 8. Wiley Online Library. 2021, pp. 17–29.

[27] Xingyue Pan et al. “Enhancing Spatiotemporal Resampling with a Novel MIS Weight”. In: *Computer Graphics Forum*. Vol. 43. 2. Wiley Online Library. 2024, e15049.

[28] Steven G Parker et al. “Optix: a general purpose ray tracing engine”. In: *Acm transactions on graphics (tog)* 29.4 (2010), pp. 1–13.

[29] Georg Petschnigg et al. “Digital photography with flash and no-flash image pairs”. In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 664–672.

[30] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. isbn: 0128006455.

[31] Stefan Popov et al. “Probabilistic connections for bidirectional path tracing”. In: *Computer Graphics Forum*. Vol. 34. 4. Wiley Online Library. 2015, pp. 75–86.

[32] Rohan Sawhney et al. “Decorrelating restir samplers via mcmc mutations”. In: *ACM Transactions on Graphics* 43.1 (2024), pp. 1–15.

[33] Christoph Schied et al. “Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination”. In: *Proceedings of High Performance Graphics*. 2017, pp. 1–12.

[34] Martin Šik et al. “Robust light transport simulation via metropolised bidirectional estimators”. In: *ACM Trans. Graph* 35.6 (2016), p. 245.

[35] Justin F Talbot. *Importance resampling for global illumination*. Brigham Young University, 2005.

[36] Yusuke Tokuyoshi. “Efficient Spatial Resampling Using the PDF Similarity”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.1 (2023), pp. 1–19.

[37] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

[38] Eric Veach and Leonidas J Guibas. “Metropolis light transport”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 65–76.

[39] Jiří Vorba et al. “On-line learning of parametric mixture models for light transport simulation”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–11.

[40] Ingo Wald et al. “Embree: a kernel framework for efficient CPU ray tracing”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–8.

[41] Chris Wyman and Adam Marrs. “Introduction to directx raytracing”. In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs* (2019), pp. 21–47.

[42] Chris Wyman and Alexey Panteleev. “Rearchitecting spatiotemporal resampling for production”. In: *Proceedings of the Conference on High-Performance Graphics*. 2021, pp. 23–41.

[43] Chris Wyman et al. “A gentle introduction to ReSTIR path reuse in real-time”. In: *ACM SIGGRAPH 2023 Courses*. 2023, pp. 1–38.

[44] Dmitry Zhdan. *Fast denoising with self stabilizing recurrent blurs*. 2020.

A

ReSTIR+ Reservoir Combinations

The following are the algorithms for combining multiple reservoirs in a similar vein to [3] using WRS with subset division. They are analoguous to 'Algorithm 4: Combining the streams of multiple reservoirs' and 'Algorithm 6: Unbiased combination of multiple reservoirs' respectively. We use the reservoir data structure from Algorithm 5 and use $r.W_i$ to denote the UCW associated with sample i in reservoir r. We denote the number of output samples as N.

Algorithm 6 Biased Reservoir Combination - ReSTIR+

function combineReservoirsBiased($\hat{q}$, $r_1, r_2, ..., r_k$) ▷ Target function of the canonical pixel and the k reservoirs to combine
    Reservoir s
    for each $r \in \{r_1, r_2, ..., r_k\}$ **do** ▷ Process each reservoir
        for $i \leftarrow 1$ **to** N **do** ▷ Process each sample contained in the reservoir
            s.update($r.Y_i$, $\hat{q}(r.Y_i) \cdot r.W_i \cdot r.c_i$, $r.c_i$)
        end for
    end for

    for $i \leftarrow 1$ **to** N **do**
        $s.W_i \leftarrow \frac{1}{\hat{q}(s.Y_i)} \cdot \frac{1}{s.c_i} \cdot s.w_{sum_i}$
    end for
    return s
end function

Algorithm 7 Unbiased Reservoir Combination - ReSTIR+

function combineReservoirsUnbiased($\hat{q}$, $r_1, r_2, ..., r_k$, $\hat{q}_1, \hat{q}_2, ..., \hat{q}_k$) ▷ Target function of the canonical pixel, the k reservoirs to combine, and the k target functions associated with each of the input reservoirs
    Reservoir s
    for each $r \in \{r_1, r_2, ..., r_k\}$ **do** ▷ Process each reservoir
        for $i \leftarrow 1$ **to** N **do** ▷ Process each sample contained in the reservoir
            s.update($r.Y_i$, $\hat{q}(r.Y_i) \cdot r.W_i \cdot r.c_i$, $r.c_i$)
        end for
    end for

    $Z \leftarrow 0^N$ ▷ Adjusted confidence weights for each of the sub-reservoirs
    for $i \leftarrow$ **to** k **do** ▷ Loop over all reservoirs and their associated target functions
        for $j \leftarrow 1$ **to** N **do** ▷ Loop over all sub-reservoirs
            if $\hat{q}_i(s.Y_j) > 0$ **then**
                $Z_j \leftarrow Z_j + r_i.c_j$
            end if
        end for
    end for

    for $i \leftarrow 1$ **to** N **do**
        $s.W_i \leftarrow \frac{1}{\hat{q}(s.Y_i)} \cdot \frac{1}{Z_i} \cdot s.w_{sum_i}$
    end for
    return s
end function

B

Source Code

The source code and scene data used to generate the presented results can be found at the following URLs:

- **R-OMIS Implementation**: Implementation of the presented estimators and their associated techniques, namely ReSTIR+, R-MIS, and R-OMIS.
  `https://github.com/MrMagnifico/romis`
- **Baseline ReSTIR Implementation**: Implementation of ReSTIR as described in [3] with no algorithmic alterations.
  `https://github.com/MrMagnifico/cpp-restir`
- **Visualisations**: Python code used to construct many of the presented diagrams, graphs, and visualisations.
  `https://github.com/MrMagnifico/romis-visuals`

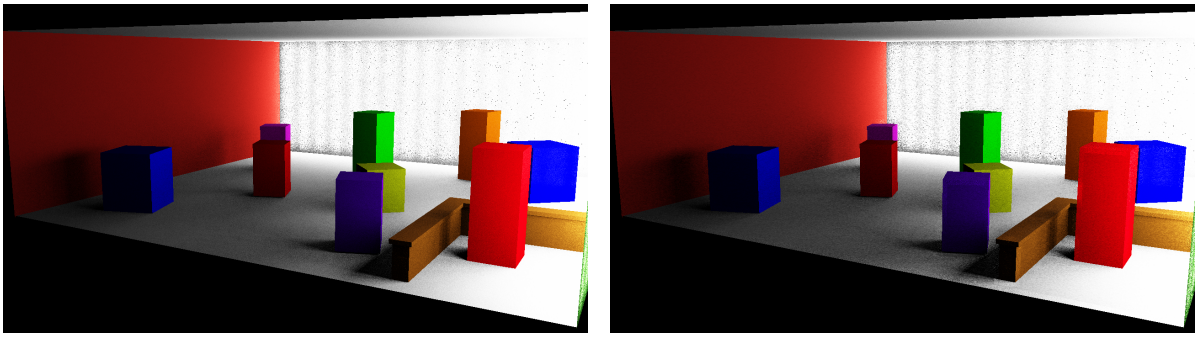
C

Full Images

This appendix contains the full images used to composite the striped comparisons and MAPE heatmap comparisons in chapter 5.

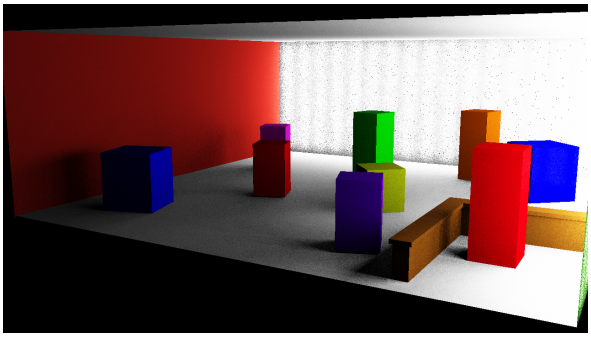
C.1. Similarity Heuristics

C.1.1. Final Renders

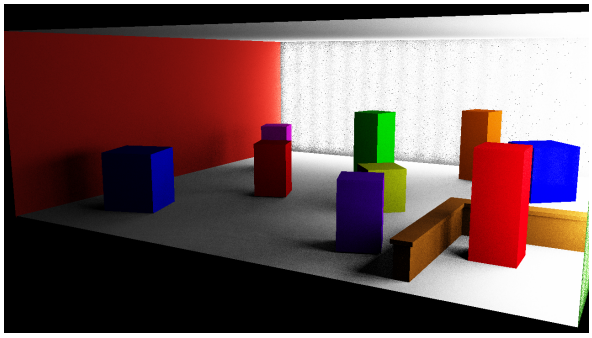


(a) Similar

(b) Dissimilar



(c) Equal

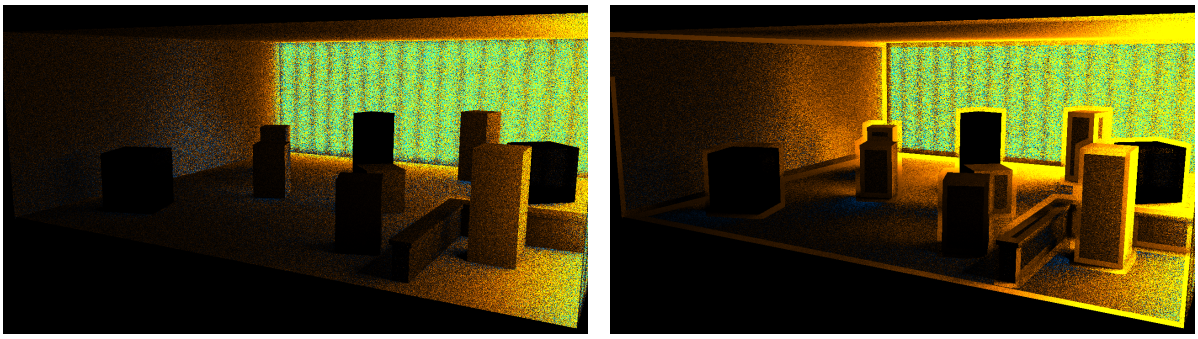


(d) Random

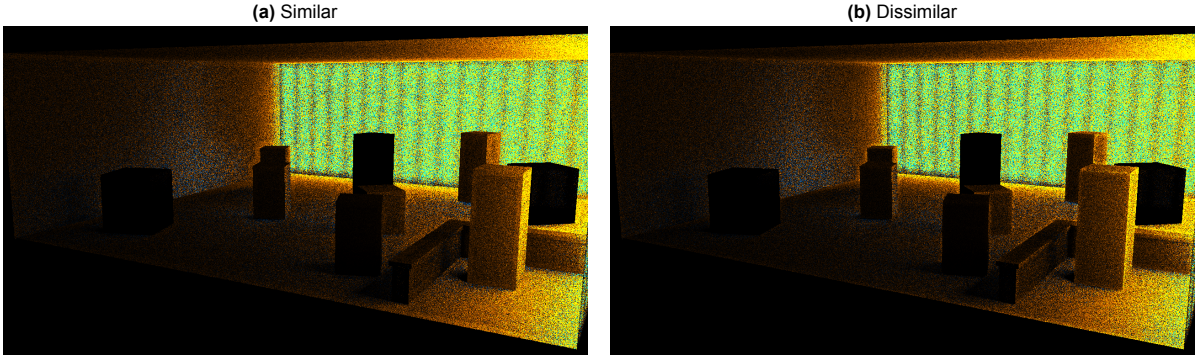
Figure C.1: Similarity Heuristics - Final Renders

C.1.2. Alpha Vectors

Canonical



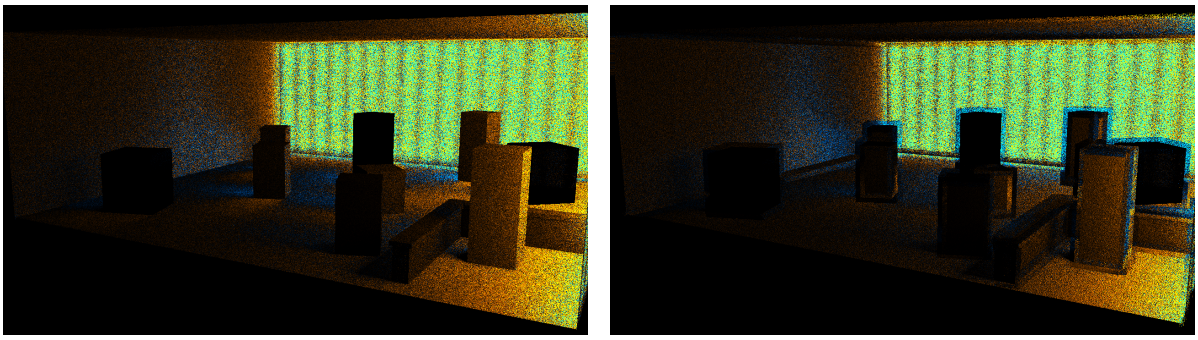
(a) Similar **(b)** Dissimilar



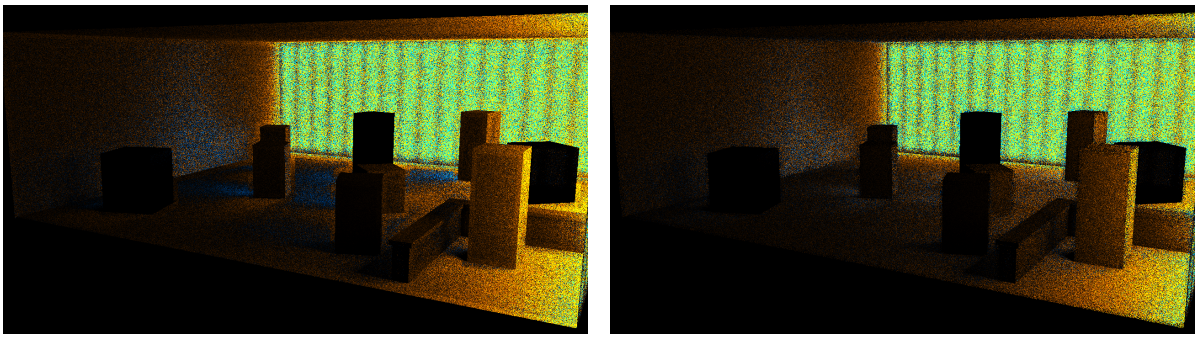
(c) Equal **(d)** Random

Figure C.2: Similarity Heuristics - Canonical Alpha Vectors

Neighbour



(a) Similar **(b)** Dissimilar

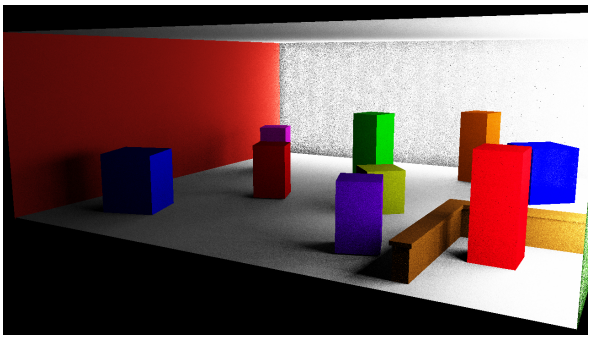


(c) Equal **(d)** Random

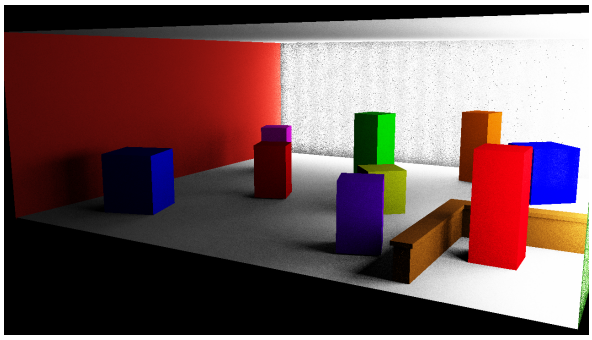
Figure C.3: Similarity Heuristics - Neighbour Alpha Vectors

C.2. Canonical Candidate Count

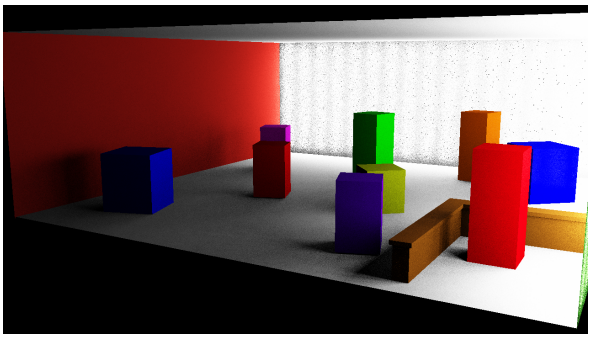
C.2.1. Final Renders



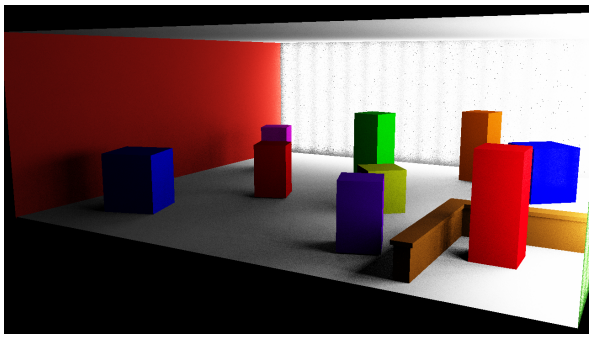
(a) 16 canonical candidates



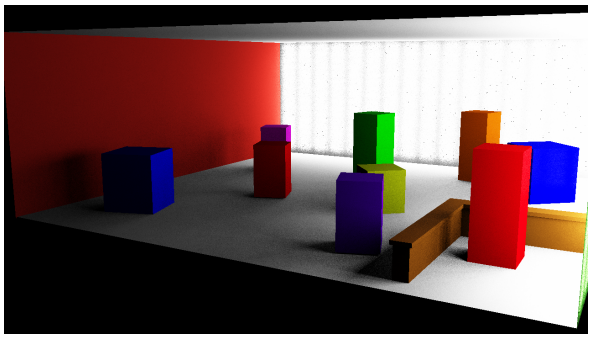
(b) 32 canonical candidates



(c) 64 canonical candidates



(d) 128 canonical candidates

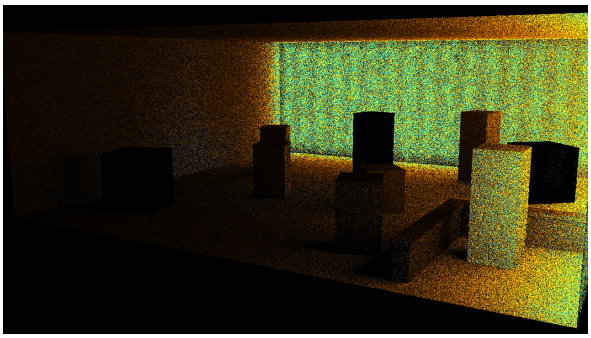


(e) 256 canonical candidates

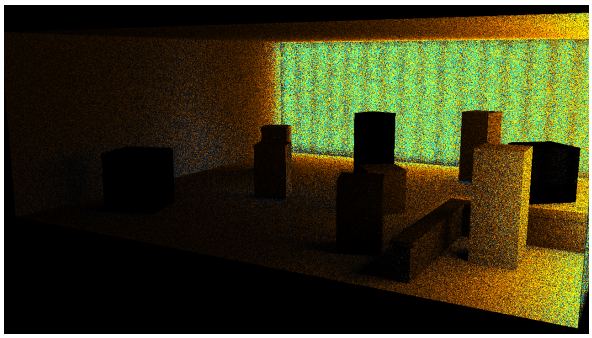
Figure C.4: Canonical Candidate Count - Final Renders

C.2.2. Alpha Vectors

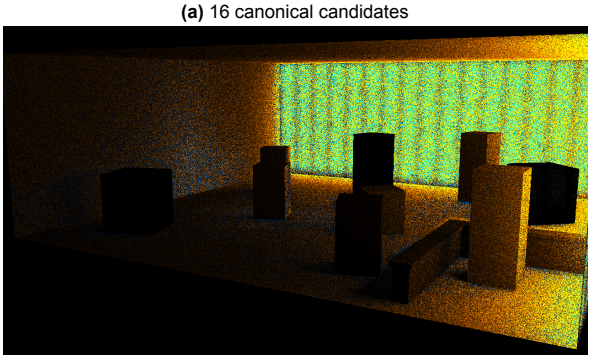
Canonical



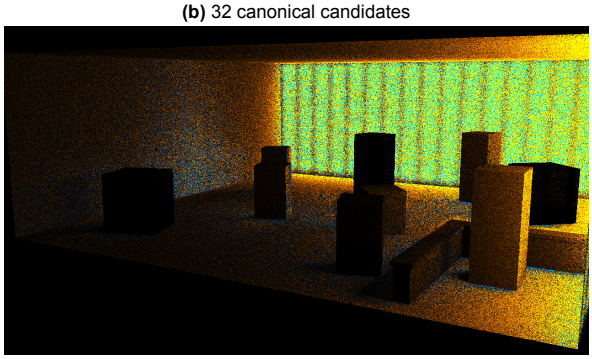
(a) 16 canonical candidates



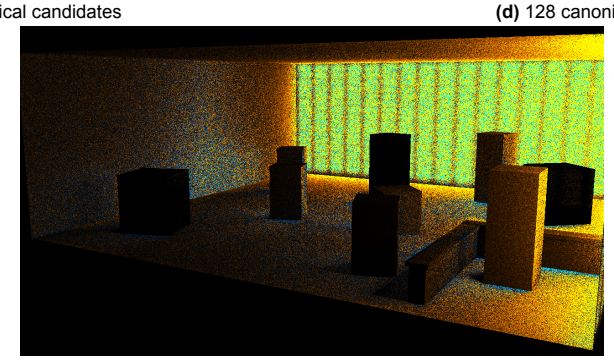
(b) 32 canonical candidates



(c) 64 canonical candidates



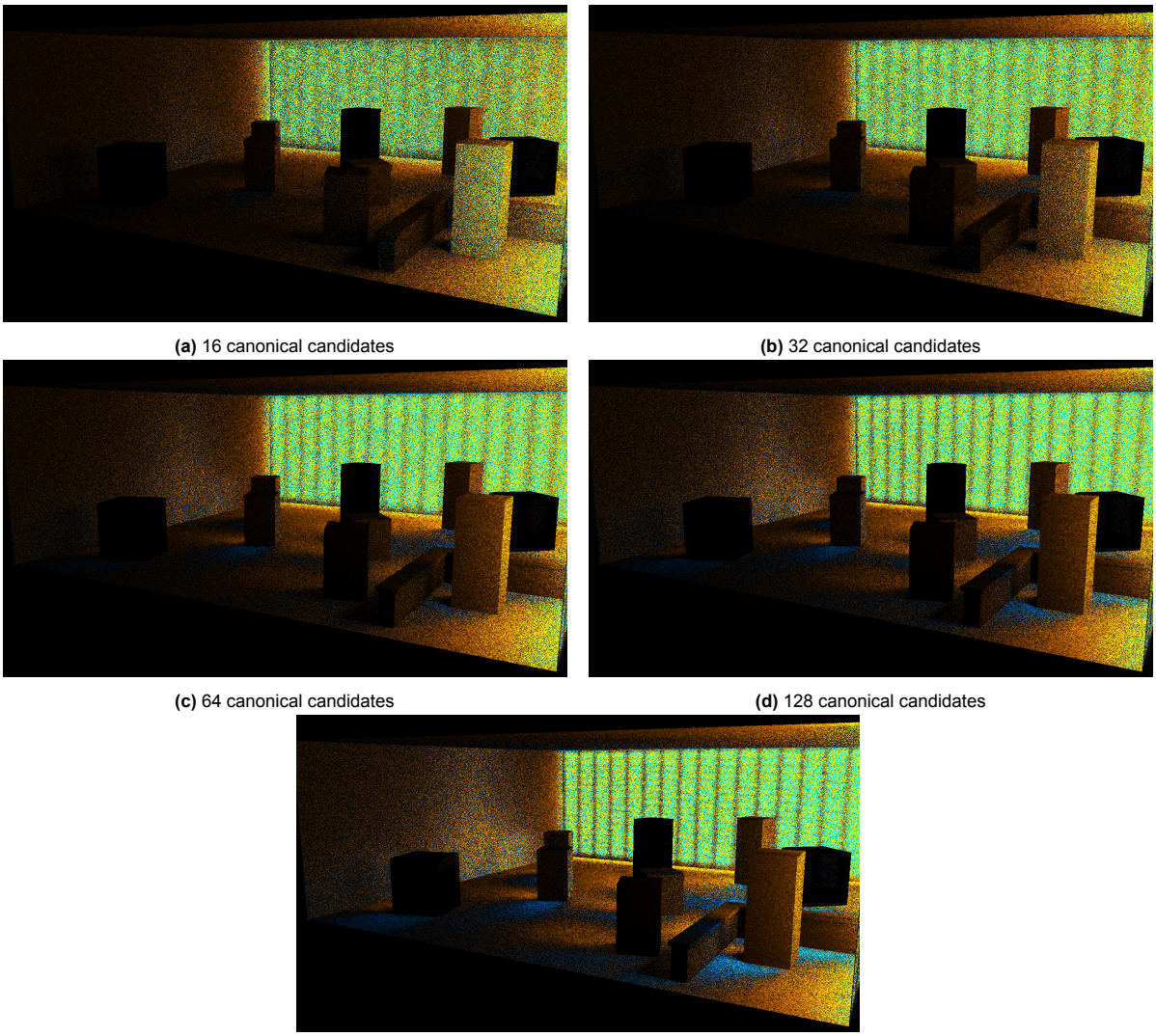
(d) 128 canonical candidates



(e) 256 canonical candidates

Figure C.5: Canonical Candidate Count - Canonical Alpha Vectors

Neighbour



(a) 16 canonical candidates

(b) 32 canonical candidates

(c) 64 canonical candidates

(d) 128 canonical candidates

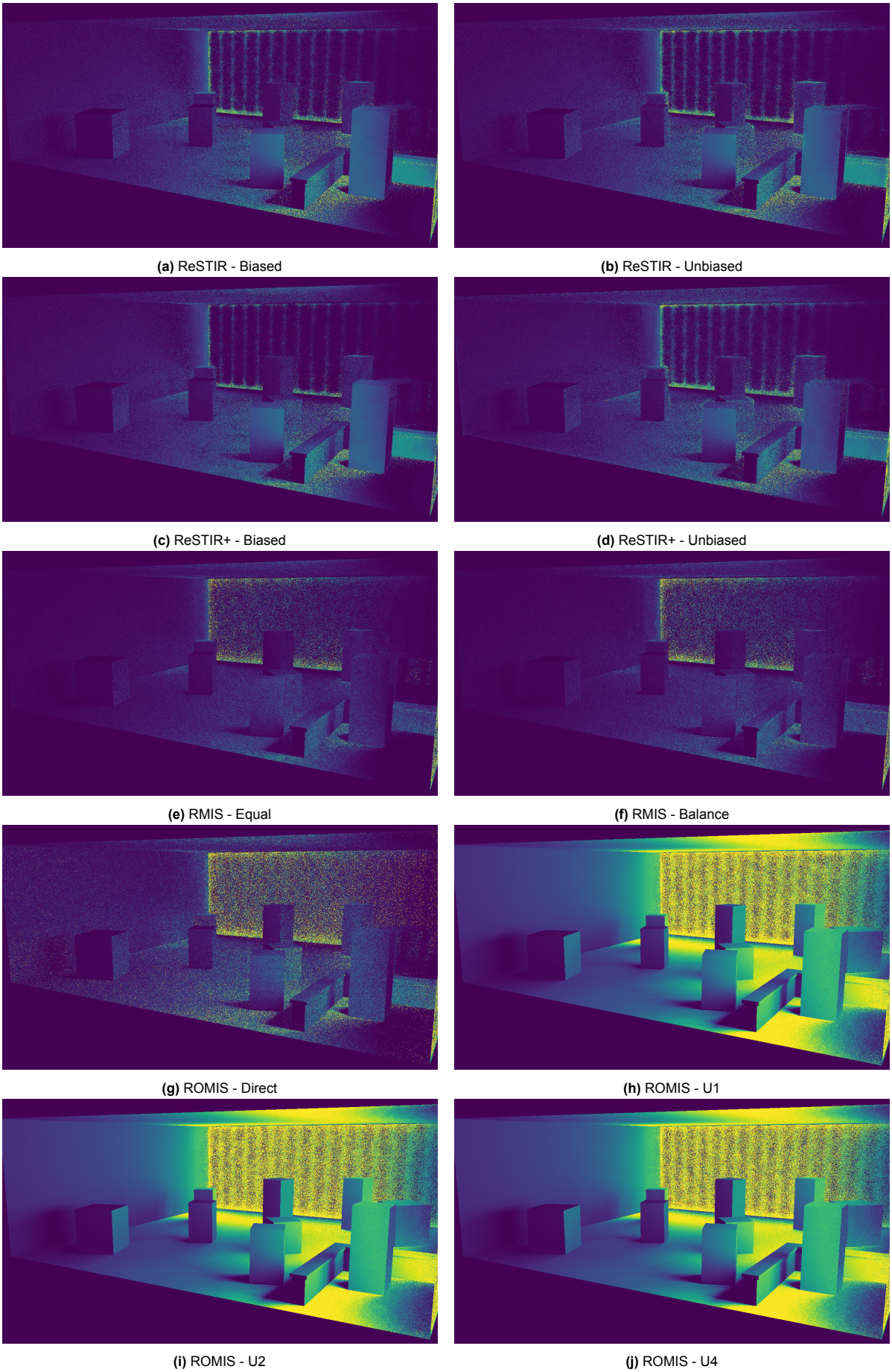
(e) 256 canonical candidates

Figure C.6: Canonical Candidate Count - Neighbour Alpha Vectors

C.3. MAPE Heatmaps

Heatmaps corresponding to renders of all tested scenes (see Section 5.1.2 and Section 5.1.3) are provided for both a single iteration and 5 iterations. The presented configurations correspond to those detailed in Section 5.5.

C.3.1. 1 Iteration



(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

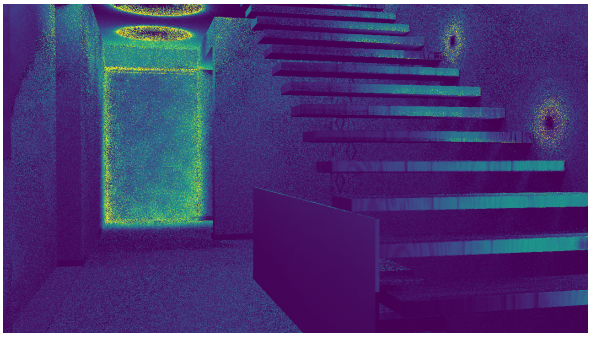
(g) ROMIS - Direct

(h) ROMIS - U1

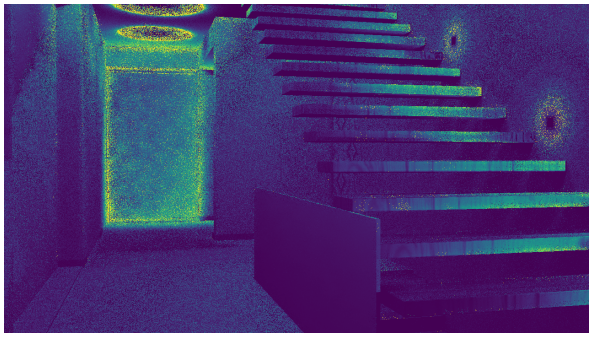
(i) ROMIS - U2

(j) ROMIS - U4

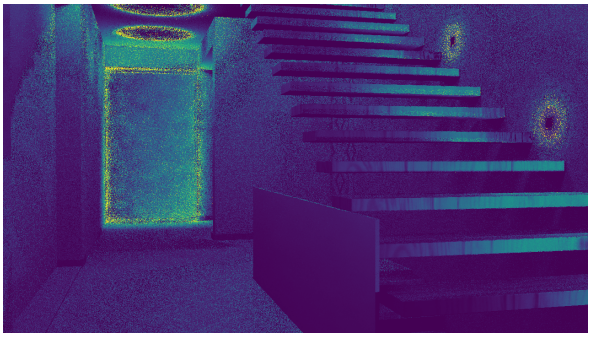
Figure C.7: Cornell Nightclub



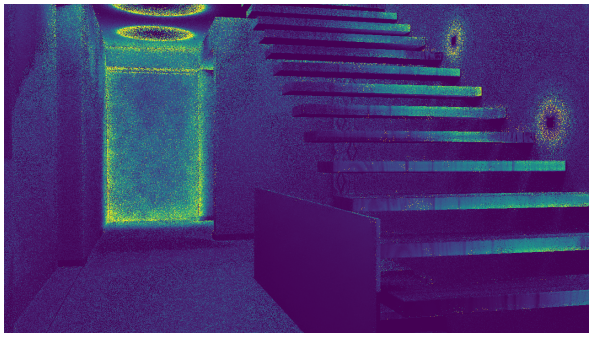
(a) ReSTIR - Biased



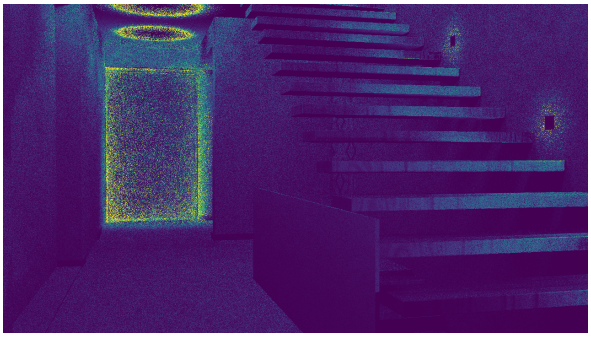
(b) ReSTIR - Unbiased



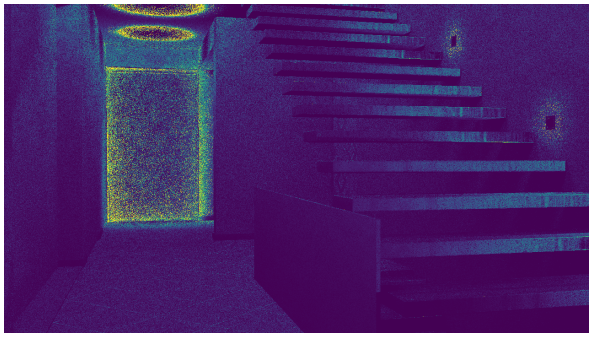
(c) ReSTIR+ - Biased



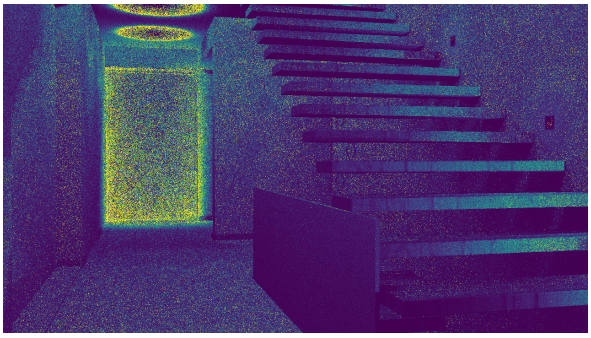
(d) ReSTIR+ - Unbiased



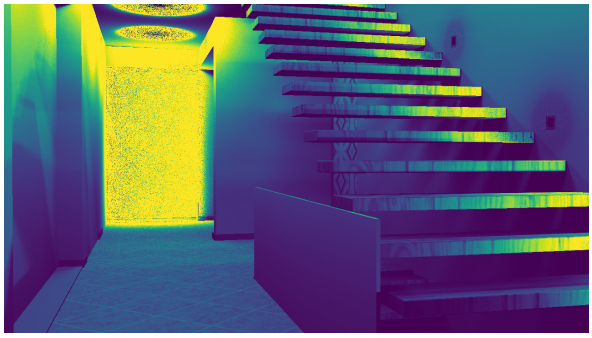
(e) RMIS - Equal



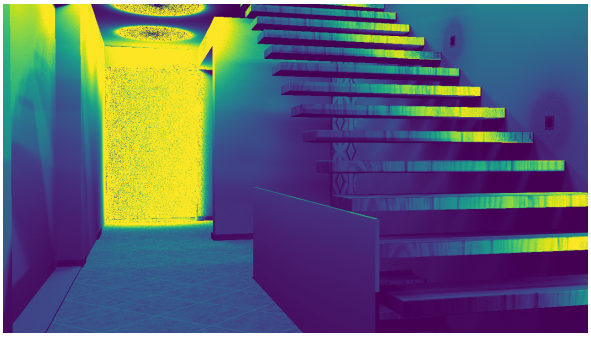
(f) RMIS - Balance



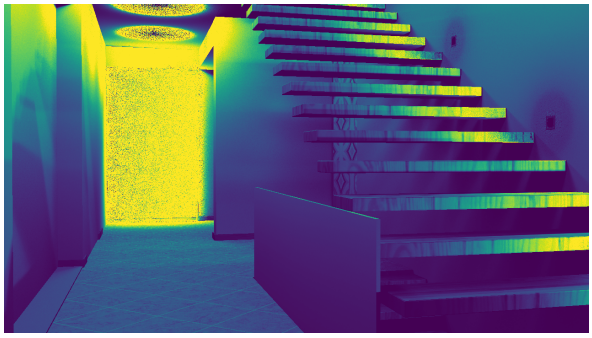
(g) ROMIS - Direct



(h) ROMIS - U1

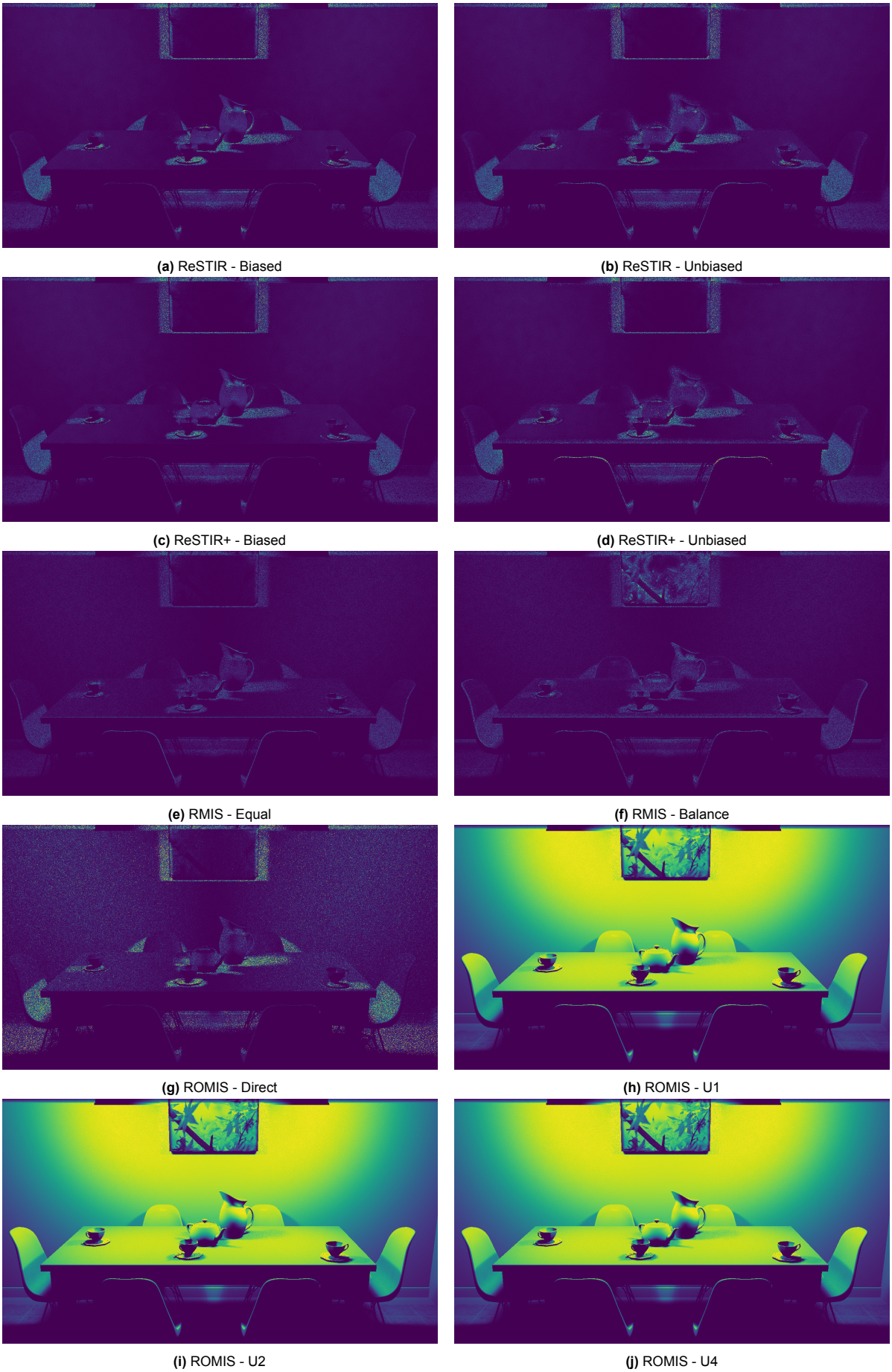


(i) ROMIS - U2



(j) ROMIS - U4

Figure C.8: Modern Hall



(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

(g) ROMIS - Direct

(h) ROMIS - U1

(i) ROMIS - U2

(j) ROMIS - U4

Figure C.9: The Breakfast Room

(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

(g) ROMIS - Direct

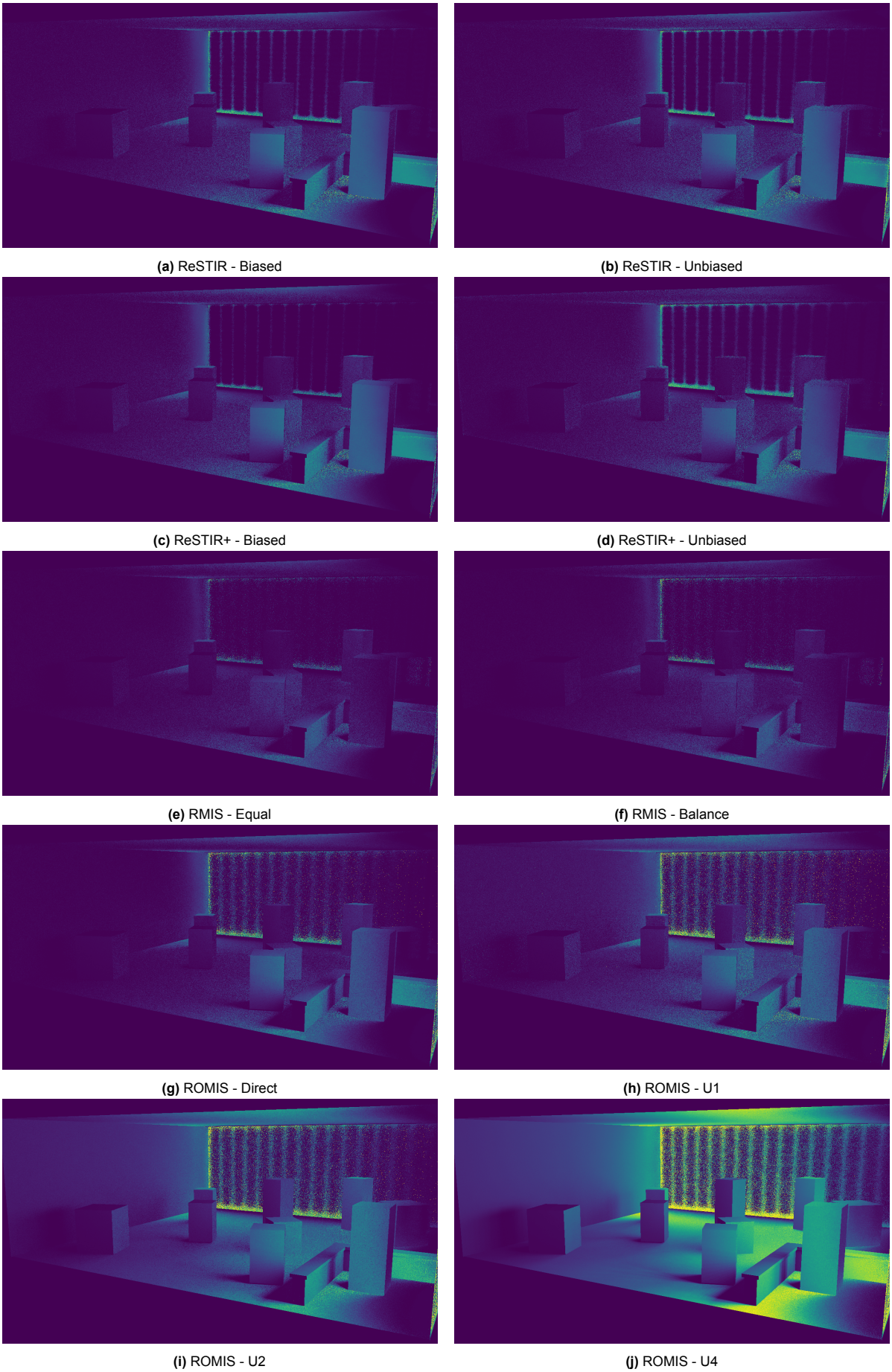
(h) ROMIS - U1

(i) ROMIS - U2

(j) ROMIS - U4

Figure C.10: The Modern Living Room

C.3.2. 5 Iterations



(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

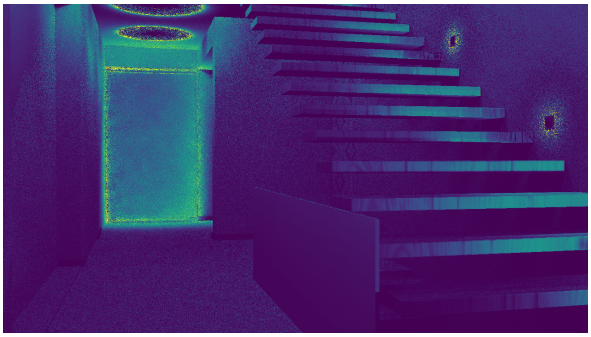
(g) ROMIS - Direct

(h) ROMIS - U1

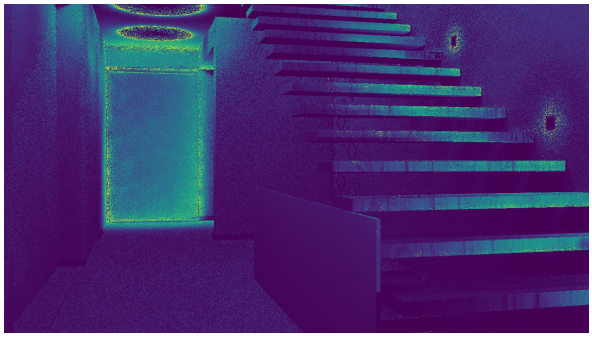
(i) ROMIS - U2

(j) ROMIS - U4

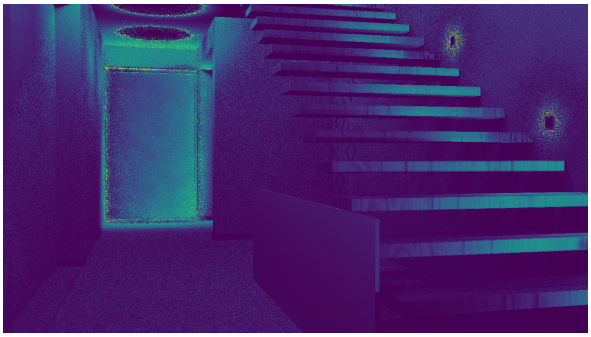
Figure C.11: Cornell Nightclub



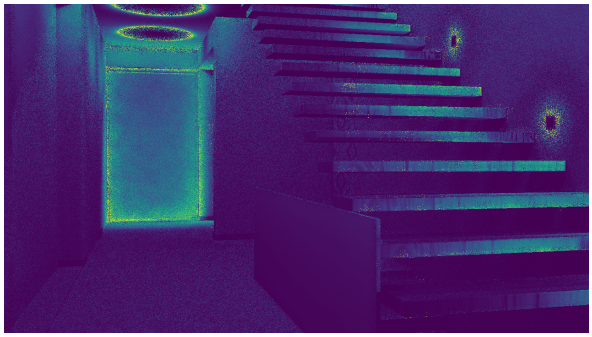
(a) ReSTIR - Biased



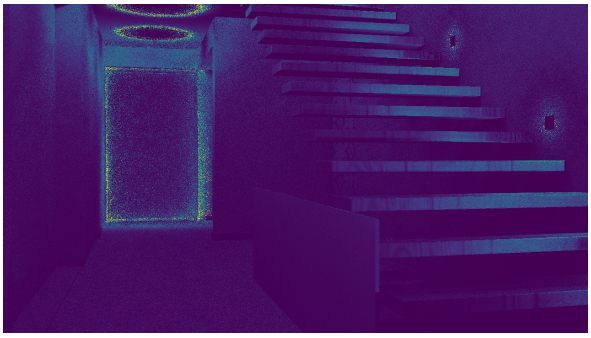
(b) ReSTIR - Unbiased



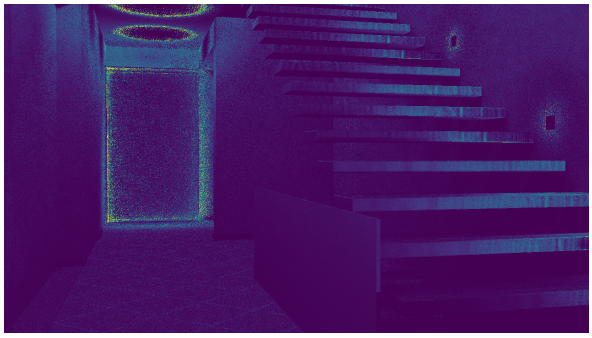
(c) ReSTIR+ - Biased



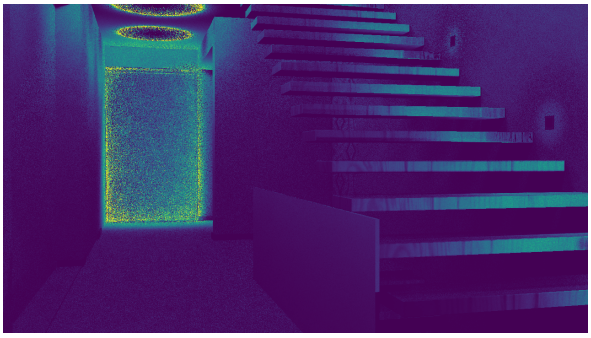
(d) ReSTIR+ - Unbiased



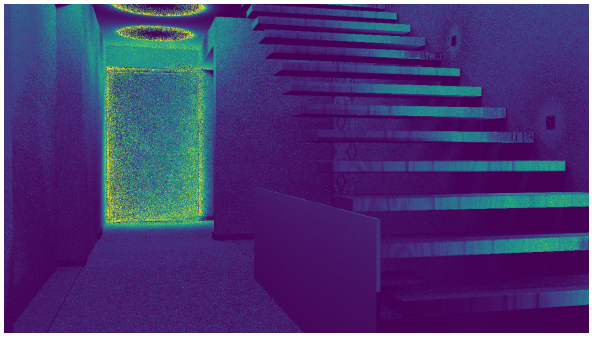
(e) RMIS - Equal



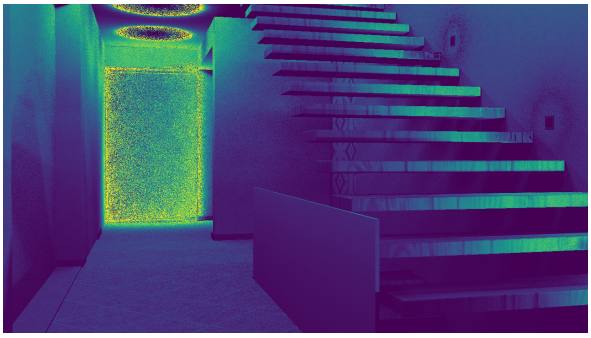
(f) RMIS - Balance



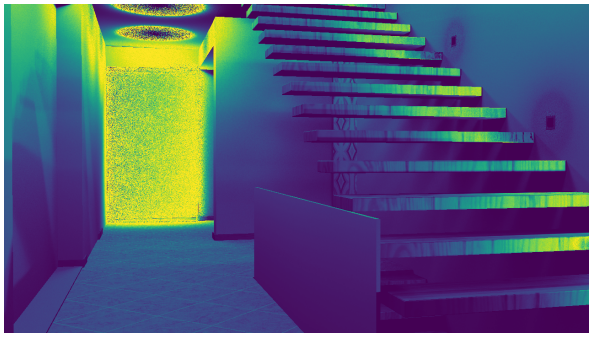
(g) ROMIS - Direct



(h) ROMIS - U1

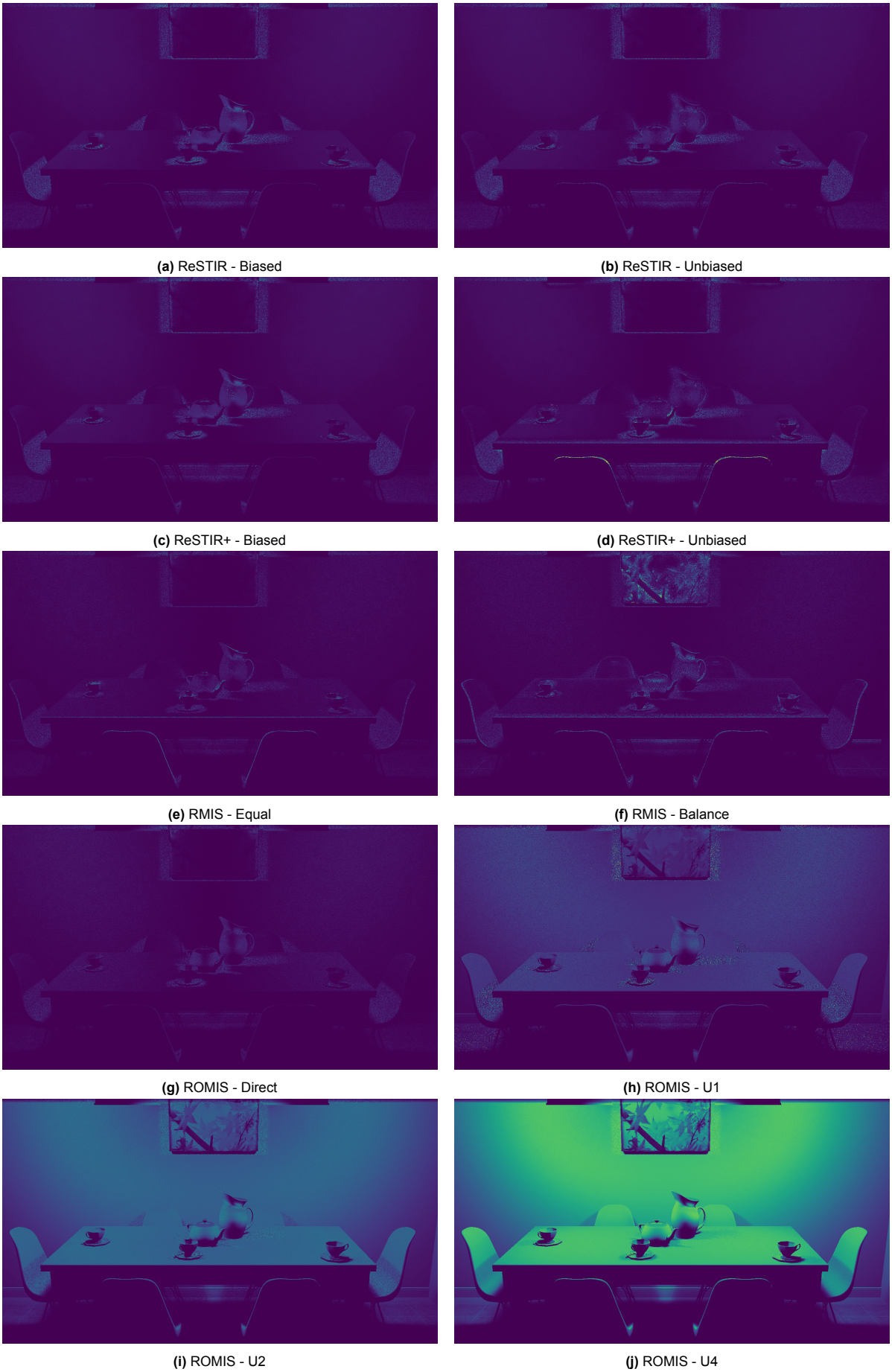


(i) ROMIS - U2



(j) ROMIS - U4

Figure C.12: Modern Hall



(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

(g) ROMIS - Direct

(h) ROMIS - U1

(i) ROMIS - U2

(j) ROMIS - U4

Figure C.13: The Breakfast Room

(a) ReSTIR - Biased

(b) ReSTIR - Unbiased

(c) ReSTIR+ - Biased

(d) ReSTIR+ - Unbiased

(e) RMIS - Equal

(f) RMIS - Balance

(g) ROMIS - Direct

(h) ROMIS - U1

(i) ROMIS - U2

(j) ROMIS - U4

Figure C.14: The Modern Living Room