CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A study of parameter trade-offs with Re-STIR for low-cost rendering and high-quality reflections

Master's thesis in Computer science and engineering

YUHAN WANG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

A study of parameter tradeoffs with ReSTIR for low-cost rendering and high-quality reflections

YUHAN WANG

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

A study of parameter trade-offs with ReSTIR for low-cost rendering and high-quality reflections
YUHAN WANG

Supervisor: Erik Sintorn, Department of Computer Science and Engineering
Examiner: Ulf Assarsson, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

A study of parameter trade-offs with ReSTIR for low-cost rendering and high-quality reflections

YUHAN WANG
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

It is the aim of this thesis to conduct a thorough analysis related to the effects of parameters on render efficiency or image quality of Path Tracing algorithm known as ReSTIR. In this case, the paper is based on the prior algorithms like ReSTIR, ReSTIR GI, and ReSTIR PT, and provides an integrated viewpoint. The analysis begins by first clearly explaining the ReSTIR algorithm before conducting an extensive examination regarding its multiple parameters influencing it provided speed. As a result, considerable exchanges between/among these parameters and rendering modes are bound to occur. So, the goal is to balance the efficiency gains and quality of the images produced with a ray tracing supported GPU. In this case, therefore, we have employed plots in determining the best values for parameters that dramatically affect the efficiency and image quality of ReSTIR real time rendering. These are all beneficial in mapping which parameters need tuning in order to yield specific renderings. In conclusion, this thesis provides a clear insight about the influence of parameters in ReSTIR (PT) at render efficiency and quality. Such recommendations suggest improvement strategies for different facets of the algorithm, which offer prospects for increasing the runtime rendering speed and the quality of images in ReSTIR-based systems.

Keywords: Computer, graphics, computer graphics, global illumination, ReSTIR, ray tracing.

Acknowledgements

First and foremost, I would like to express my gratitude to my examiner Ulf Assarsson for his support throughout the duration of the thesis. He supported me in every aspect of my endeavor throughout my stay: He let me decide on the subject of my thesis without restrictions, provided me with valuable and profound feedback regarding thesis topics, allowed me to use the laboratory in case I needed equipment for certain experiments, and guided me throughout the decision-making process of my research direction. I should thank my examiner for his assistance since I could not have completed this thesis project without his support.

I also want to acknowledge my supervisor, Erik Sintorn, who pay much of his time and effort with focus and care on making sure that my dissertation work is good enough. This man also came forward to make very good comments and suggestions when I was relating the project.

Concerning this, I would like to thank my opponents, Andreas Rudén and Ludvig Andersson and the audience during the presentation as well. He did not interrupt continuously elaborate his questions and mine them, answering them; rather he helped me deeply think more inspiringly and look at my completed work in multiple ways, giving me more probable possibilities in creation.

Last but not the least, I would like to extend my profound gratitude to my mom, Xiuyun Wang, for helping me in accessing my home in China over internet which is thousands of miles away. Therefore, I could perform most of the experiments using the PC at home because I was sometimes unable to use the laboratory because of time restriction and performance of the equipment. Even though the process of experiment was very tedious, it was useful a lot when dealing with this project. Moreover, the encouragement and moral support which my mother has been providing to me since my elementary school period and her consummate love for me also been a vital force for propelling me towards it until now.

Yuhan Wang, Gothenburg, 2024-06-10

Contents

List of Figures

1

Introduction

The field of computer graphics is as old as several decades now, but it is only in the last few decades that it has matured. The subject of computer graphics developed in the early 1960s when scientists began searching for the method of image rendering and visualization by using computer equipment. Ivan Sutherland designed Sketchpad in 1963 a system which allowed users to draw and manipulate graphical figures with light pen[1]. This became the first tool for interactive computer graphics. On the other hand the sphere of global illumination based on the concept of ray tracing begun in early 1960s with the work of Arthur Appel[2]. Ray tracing is the process of calculating the paths of light rays which are projected through the scene of view and from the eye of the viewer as a fallback for reflection, refracting or indirect lighting.

During the seventies, wireframe modeling and 2D graphics were the main objects of researchers and developers interests. New graphics standards such as GKS (Graphical Kernel System) and later, PHIGS (Programmers Hierarchical Graphics System) were instrumental in supporting common API interfaces to construct 2D graphics interfaces[3][4]. Other advancements included the identification of the raster graphics displays, and the coming up with of the scanline algorithms for rendering. Although the idea of radiosity was first suggested in the late seventies and in early eighties, with efforts made towards its utilization as a means of global illumination. Radiosity is more concerned with the calculation of the energy transfer equation in a scene by taking into account of the diffuse interreflection of light sources. The radiosity method was adopted for modeling of the distant and secondary lights in large and static environments.

The year 1980 was the beginning of a higher paradigm for 3D graphics and geometric modelling. The increase of capabilities of algorithms for hidden surface removal which can represent real values, such as Z-buffer, allowed for more realistic rendering[5]. Phong shading model that was developed by Bui Tuong Phong in 1973 was also an extension to the Lambertian reflectance model and became popular for realistic representation of surface reflection and lighting[6]. Organization and advancement of the AutoCAD modeling software and the solidity modeling approaches were some of the factors that inspired the advancement and use of computer-aided design (CAD). At the same time, researchers began devising Monte Carlo methods for using global illumination, thus bringing those stochastic procedures for modeling the properties of light. Lighting techniques like path tracing, for example, use Monte

Carlo integration so that rather than calculating predefined light path contributions, path samples are randomly selected and their contributions are accumulated.

The earlier decade at the 1980s and 1990s recorded developments in the rendering methods. The technique whereby Whitted introduced in 1979 known as recursive ray tracing meant that the illumination of objects within the virtual scene could be accurately predicted by tracing rays that originated from the camera[7]. With the help of radiosity algorithms, possibilities for employing accurate global illumination effects became realistic. In 1986, James Kajiya further advanced the concepts of ray tracing by proposing that of a path tracing which is a sort of Monte Carlo integration[8]. Path tracing replicates each light path from the camera to the object and computes direct and reflected paths and indirect illumination specially specular reflection.

In 1996, Henrik Wann Jensen introduced a new approach known as the photon mappingthis makes use of global illumination effects through a two pass algorithm[9]. Although photon shooting can cast a great number of photons and obtain details of photon traversal through geometry, using a precomputed photon map to estimate indirect illumination significantly reduces the number of photons needed to achieve the desired caustics and bounces while keeping computation efficient. In the late 1990s, bidirectional path tracing was invented by Eric Veach, which forms the strengths of path tracing and photon mapping[10]. Bidirectional path tracing is more efficient, as it unlinks light path from both the virtual camera and the light sources, allowing for a more realistic simulation of interactions between light that bounces around a scene in addition to other complex transport phenomena.

Subsequently, in a number of years, the research study directed its attention towards the effectiveness and speed of global illumination algorithms. Some of these methods include progressive refinement, adaptive sampling and importance resampling that were created to offer better performance and quality by directing more computation towards the areas of firmographic data that needed it most. In 2005, Justin F. Talbot for his master thesis[11] introduced a new invented algorithm Importance Resampling for Global Illumination. It is considered the basis of many subsequent works, and the closest in terms of subject to our work is ReSTIR, described by Benedikt Bitterli et al in 2020[12]. More works for instance ReSTIR GI by Ouyang et al. 2021[13], ReSTIR PT by Daqi Lin et al. 2022[14] built on ReSTIR and aimed at exploring more prospects for obtaining or improving the ReSTIR.

Our work relies on all the related works of ReSTIR family, trying to find out if there is any way to improve the efficiency of rendering, and learn more about the tradeoff of rendering, in order to compare all the algorithms which is related with rending time, quality of images and so on.

1.1 Background

1.1.1 Global Illumination

Global lighting also termed as Global illumination is a group of methods used in computer graphics to mimic how light behaves around objects and on surfaces of a scene. It strives to create genuine and convincing lighting environments thereby taking into consideration the innate properties of light in regard to its interaction with objects and bounces off surfaces.

While using conventional techniques of computer graphics, simple models of lighting such as the Phong model or the Blinn-Phong model can be employed[15]. These models contain the direct lighting from light sources like point lights or a directional light source but it does not incorporate on indirect lighting which is luminosity that is reflected from a surface and bounces to other part of scene.

While walkthrough algorithms are focused only on trying to simulate the direct lighting, there are some other algorithms called 'Global illumination algorithms' which try to simulate both the direct and the indirect lighting effects. It includes how they interpret the contact of light on different surfaces diffusely reflecting surfaces, smoothly reflecting surfaces and transparent surfaces where light gets transmitted through objects made of translucent materials. Direct lighting is supplemented through global illumination techniques which in turn improves the render scenes taking into consideration the indirect lighting.

There are several approaches to global illumination, including:

Ray tracing: Ray tracing is a tool that calculates realistic path of a ray of light in the modeled scene. For each of the pixels on the camera, it follows the rays and then determines the point at which these rays will intersect with the objects in the scene. Global illumination can be calculated using recursive ray tracing While reflecting or refracting a ray inside the scene. However, it can be a very lengthy and unreasonably time-consuming process at times owing to the computational complexity.

Radiosity: This method is known as radiosity, as it deals with the exchange of light between several objects. It portrays the scene as numerous compartments and determines the quantity of light energy transferred between them. Due to color bleeding the indirect lighting of the scene is accurately computed using radiosity algorithms which actually solve a set of linear equations to determine the energy distribution between the surfaces.

Photon mapping: It is a technique of two operations where simulation of light is done through a technique called photon mapping where photons are traced in the scene. In the energy estimation pass, photons are emitted starting from light sources and accumulated in a data structure referred to as the photon map. In the second pass, the generated photons are then used to approximate the amount of illumination introduced at a particular point on the surface, inclusive of direct as well as bounced light.

Path tracing: For definitions and further details of path tracing, refer to the section

'Introduction to Monte Carlo Techniques'; path tracing is fundamentally the process of simulating light by tracing rays through the scene. It simplifies the complex task of the ray tracing by improving it by adding features of global illumination. Path tracing works for a ray and computes which way it goes, reflecting on other objects or passing through them and adding indirect illumination. The overall estimated solution converges to a more accurate solution as the number of samples increases; however, it uses a lot of time in computation.

Global illumination algorithms, on the other hand, attempt to capture both direct and indirect lighting effects. They take into account how light interacts with the environment, including diffuse reflection, specular reflection, and transmission through translucent materials. By considering indirect lighting, global illumination techniques enhance the realism of rendered scenes.

1.1.2 Path Tracing

Path tracing is an important rendering technique which involves the computation of radiance by following the paths of the light through a scene model and calculating how objects and surfaces within the model affect them. It is one of the most widely used, physically-realistic rendering techniques for simulating light and Rendered Global Illumination. The history of path tracing development can be traced back to several key milestones:

Theoretical Foundation: The beginning of path tracing can be traced to the work of James Kajiya who proposed the technique in 1986 in his paper[8]. The paper elucidates the algebraic coupling underlining the process of light transport through a scene and the incoming radiance detected by a camera.

Early Path Tracing Implementations: Computer graphics enthusiasts along with researchers explored the ideas of path tracing and implemented them on early computer systems during the last decades of the twentieth century and the early part of the twenty-first century. These implementations served as a verification of the theoretical ideas addressed and a proof of concept of path tracing as a plausible means of render realistic images.

Advancements in Monte Carlo Integration: Looks at the technical setup of path tracing and how path tracing utilises Monte Carlo integration methods for sampling and approximations of light sources. The advance of better algorithms for generating samples and reducing variance using importance sampling and Russian roulette made path tracing much more efficient.

Distribution Ray Tracing: Distribution ray tracing which was proposed by Eric Veach and Leonidas J. Guibas in 1995 [16][17] extended the line of thinking and used probability distributions to decide where the rays should be directed for tracing the light paths. This technique enhances path tracing by approximating the path distribution with an empirical probability distribution from the causality of the paths.

Bidirectional Path Tracing: BDPT can be credited to Henrik Wann Jensen, added

in 1996[18], this approach extended the abilities of path tracing by incorporating paths from secondary light sources (eye paths) and paths from the primary light sources (light paths). BDPT is a hybrid of both approaches and works best in those situations where the ray intersects numerous objects with different illumination and geometrical configurations.

Progressive Refinement and Real-Time Path Tracing: For a long time, researchers have invested efforts in method to render or optimize path tracing in real time interaction or in real time. The techniques like progressive photon mapping as well as progressive photon beams are rendering tools that make room for further enhancement or addition of path tracing outcome in iterative ways making the process of rendering very interactive and much faster.

Due to the fast pace of the evolution in the field of computer graphics, the researchers and developers are always in the process of certain innovations in terms of improvements and extensions of the algorithms and techniques of path tracing, which can find its application in film making, video games, visualization, etc.

1.1.3 Reservoir Resampling

Reservoir resampling, as its name suggests is a sampling technique in computer graphics and other domains in which, a set of a given number of distinct elements have to be selected out of a large set. Especially, if the size of the source set cannot be defined before or if the elements should be selected non-systematically and impartially. Reservoir sampling also makes the probability of choosing every single element that is present in the initial set uniform.

The idea here is that reservoir resampling is done to retain a reservoir initially kept as an empty one with its capability equivalent to the sample size desired. Elements are added one by one in the reservoir until all the elements of the given list have been inserted into it. Again when the reservoir is filled up, the subsequent elements substitute existing elements with a given probability where all the components are weighted equally in-order to be in the sample.

The algorithm for reservoir resampling can be summarized as follows:

Start with a reservoir of capacity k, about which the size k of the final sample is wanted. For each element encountered in the original set:

If the reservoir is not full, add the element to the reservoir in case it will be holding many such elements.

If the reservoir is full, target an element randomly, remove it from the stream and insert the new element with the probability, k/n where n is the number of elements seen up to now.

Continue this process until all elements in the original set have been processed.

Reservoir resampling means that for every element in the set, there is an equal probability that the element is selected in the final sample. This property can be potentially utilized in a number of ways, including using it to: conduct Monte Carlo

experiments, which require random sampling; derive samples that represent specific populations; or create random permutation sequences.

In relation to computer graphics application, reservoir resampling can be used in other methods such as ReSTIR where it is used to resample spaces and give estimations of indirect lighting contributions. It enables the selective control of specific light paths that should be retained when generating images and also serves to control noise and variance in the images.

1.1.4 Shift mapping

In path reuse algorithms pixel color is evaluated according to reusing of path samples with pixel with help of other pixels. Similar to ReSTIR and RIS, the formulation that Bekaert et al.[19] provides also has the paths and the samples transformed according to the same domain transform matrix at some point of the construction of their approximation equation. Meanwhile, some later work[20] is opening to various integration domains, as well as explicitly defines shift mappings to map paths between them. However, this better reuses the complex light transport paths such as specularity of glass and mirrors.

Shift mappings were first used in gradient-domain rendering, used in image reconstruction [21] [22] where the image is reconstructed in terms of discrete image gradients which are computed by subtracting a path contribution from its copies shifted into neighboring pixel. Many shift mappings have been proposed: continuing the first rough vertex [22], manifold exploration shifts [22] and half-vector copying [21] for specular transport, random number replay [21] [23] [24] and many more important additions, including bidirectional path tracing [25], photon mapping [26][27], for participating media [26], the connection and merging of vertices [28], as well as spectral rendering [29]. Our literature review revealed that shift mappings and gradient-domain rendering are better understood with the recent survey by Hua et al. from 2019. Another related idea is path perturbations and shift mappings used in Metropolis Light Transport [30][31]and local QMC exploration[32].

1.2 Aims

Here, a list of problems that the primary focus of this thesis is going to be working on can be seen: In this paper, we will determine the competency and speed of the ReSTIR algorithm within the utilitarian low-cost rendering paradigm that is imperative for real-time or interactive rendering. It is also possible to analyze the changes in different configurations of ReSTIR when regarding rendering quality and performance, particularly as to reflections in the scene.

The following parameters of ReSTIR will be explored for their ability to impact the accuracy and rate of convergence as applied to the estimation of indirect light, while keeping in mind the level of precision needed for rendering realistic lighting. One other approach is to come up with ways to identify the best approach to set the parameters of ReSTIR so that the reflections are as accurate as possible while

at the same time overcoming the number of rays, spatial, and temporal rates of re-sampling that is used.

Furthermore, we can match ReSTIR with other total reflection techniques like basic path tracing or screen-space reflection approaches based on the visualization quality, computational complexity, and importance for scenes with complicated reflections. It is also useful to offer practical suggestions and rules for the parameter choices in ReSTIR to pick an optimal level of rendering quality in low-cost rendering approaches and the maximal speed of image computations.

1.3 Questions

One of the questions of this thesis is to find out several parameter values for real-time ReSTIR techniques that can efficiently handle dynamic lighting and reflections. This problem necessitates a fundamental knowledge of the ReSTIR concepts alongside finding new approaches to deal with real-time issues that may be encountered during the process. The task comprises designing and testing the algorithm on which the technique is based through a prototype.

Because often the problem of global illumination is quite comprehensive, certain aspects of the problem might have to be omitted by this thesis. It will be centered mostly around tweaking settings to achieve an optimum and usable realtime GI mode, and not every feature, while things like high-detail caustics might be omitted.

2

Related Work

2.1 ReSTIR

ReSTIR is a real-time global illumination algorithm that seeks to estimate and accurately compute pixel-importance-based light transport for dynamic and interactive scenes. It was initially proposed as a research paper by NVIDIA researchers Benedikt Bitterli et al., in 2020 and has attracted attention for its capability to efficiently provide high-quality GI solutions in real-time. [12]

Still modern techniques like path tracing might be time-consuming, especially in scenes where objects and lights are in constant motion or when changes in the scene are to be rendered in real-time. ReSTIR addresses this challenge by combining ideas from two different techniques: ordinally correlated /discriminated resampling (from spatiotemporal resampling); and resampled importance sampling.

ReSTIR looks much more attractive because its crucial benefit is that it performs well in terms of dynamic scenes. Here the computation is performed only on the important paths while the least important paths are sampled using a very basic data structure called reservoir and thus it does not take hours to render a frame but it is real-time capable and generates beautiful looking global illumination effects. However, ReSTIR is only the basis of many ReSTIR algorithms, for example, ReSTIR GI[13], ReSTIR PT[14] etc. And its implementation and performance, which are discussed above, might be different in one or other scene and depending on different choices.

2.2 ReSTIR GI

This method called ReSTIR GI was developed by Ouyang et al. in 2021 [13]. The algorithm introduced in this paper is a dedicated path-sampling method that can be used for indirect lighting and is well suited for highly parallel GPU architectures. While ReSTIR proposes an algorithm for the spatial and temporal resampling in the screen space, it is heavily based on the resampling of multi-bounce indirect light paths, which are computed through path tracing. This method enables dissemination of useful information concerning excessive or scarce supply of lighter points as a function of time and as a function of the pixel coordinates in the picture. Therefore, with using this algorithm, a large number of possible errors that arise from

more common path-tracing techniques can be neglected. Strikingly, ReSTIR GI showcases those improvements with merely one sample per pixel per frame in the different test scenes, where the decrease in the mean squared error (MSE) varies between 9.3x and 166x. In addition, when utilized in conjunction with denoiser, it allows for the creation of accurate path-traced global illumination at real-time frame rates on modern GPUs.

2.3 ReSTIR PT

ReSTIR PT (Reservoir-based Spatial-Temporal Importance Resampling Path Tracing) is a kind of further advanced global illumination method that uses the advantage of the importance sampling in the efficiency while using the advantage of the path tracing in precision. It aims at rendering globally illuminated scenes in real-time with high photorealism by employing global illumination techniques.

Standard methods of configuring path tracing, while very effective, are often time-consuming, particularly in instances where numerous lamps and other objects are arranged in complex configurations. ReSTIR PT presents a reservoir-based method that samples and resamples light paths depending on their importance, which is essential for efficient computation and real-time usage.

When importance sampling, which funnels computations into important path space, is integrated with path tracing, which closely models light transport in scenes, it yields interactive-quality global illumination in interactive ReSTIR PT. It is most effective and efficient in scenarios such as real-time rendering and small-unit gaming and simulations that require frequent human interactivity.

2.4 Other Work

However, there are also some other research works on ReSTIR and several other related works of the direction. In 2021, Guillaume Boissé[33] proposed a technique of estimating the importance sampling when rendering light sources from arbitrary vertices along the eye path in the spatiotemporal domain, which is also based on reservoirs but operates in the world space. The structure of this specific multinomial distribution allows for the stochastic reuse of neighboring reservoirs in space and time while making efficient resampling of reservoirs across the time-space domain at any spatial location easily achievable.

In more detail, in 2022, Xander Hermans formulated Voxel ReSTIR, an algorithm inherited from ReSTIR as proposed herein was designed to work with voxel worlds and experimented with ReSTIR[34]. The experimentation was done intensively, by the author of this paper, to assess the efficacy of utilizing various algorithms in development given a range of environments to work under. Those results spoke for themselves and clearly demonstrated that both the point light and area light algorithms presented by the author outperformed ReSTIR in terms of both speed and image quality when the two packages were tested under the same conditions.

More surprisingly, the authors algorithm is nicely fit for the GPU implementation, therefore, it has the potential to reach near real-time performance with consumer hardware.

In the same year, another algorithm, namely Decorrelating ReSTIR Samplers via MCMC Mutations, was proposed by Sawhney et al.[35] Researchers proposed this algorithm after they had realized that the efficiency of the ReSTIR algorithm had some gaps. Notably, the random shedding of cross samples often led to gross correlation distortions if the spatial and temporal dimensions were not controlled, and premier reservoirs that would contain many samples attracted writeln deficiencies due to duplicate samples. To overcome these issues, the authors propose an efficient method of interleaving Markov Chain Monte Carlo(MCMC) mutations with reservoir resampling. This was especially useful in handling glossy materials as well as tough lighting in several scenes of a video. This is one of the biggest strengths of this method since there is no possibility of any sort of bias being interjected into the process. The authors concluded that important improvements in image quality can be demonstrated in practical applications when just a single mutation per reservoir sample in each frame of the sequence was incorporated.

3

ReSTIR Algorithm

3.1 Preliminaries

The following equation is the basis of the ray tracing algorithm:

$$L(y,\omega) = \int_A \rho(y, y\vec{x} \leftrightarrow \vec{\omega}) L_e(x \leftarrow y) G(x \leftrightarrow y) V(x \leftrightarrow y) dA_x \tag{3.1}$$

L represents the reflection radiance. Its two parameters y and w mean the light source point and outgoing radiance direction. ρ is commonly used to be BSDF, L_e is the emitted radiance, V is the visibility term between x and y, and G is the geometry term containing inverse squared distance and cosine terms.

Importance sampling(IS) Importance sampling uses the standard Monte Carol sampling method. It estimates an integral by choosing N random samples x_i from a source PDF $p(x_i)$ to compute the following equation:

$$\langle L \rangle_{is}^N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\hat{p}(x_i)} \approx L \tag{3.2}$$

Ideally, we have p(xi) proportional to f(xi) in order to reduce variance.

Multiple Importance Sampling

3.2 Resampled Importance Sampling(RIS)

Resampled Importance Sampling(RIS) was first introduced by Talbot in 2005[11]. It samples approximately proportional to the product of some of the terms. RIS generates $M1$ candidate samples $x = \{x_1, ..., x_M\}$ from a source distribution p that is sub-optimal. An index $z \in \{1, ..., M\}$ is randomly chosen from this pool of candidates with discrete probabilities:

$$p(z|x) = \frac{w(x_z)}{\sum_{i=1}^{M} w(x_i)} \quad \text{and} \quad w(x) = \frac{\hat{p}(x_i)}{p(x_i)} \tag{3.3}$$

Where $\hat{p}(x)$ is the target probability density function (PDF) for which no direct sampling algorithm may exist, a sample y is selected and used to calculate the 1-sample RIS estimator. The 1-sample RIS estimator is computed as follows:

$$\langle L \rangle_{ris}^{1,M} = \frac{f(y)}{\hat{p}(y)} \cdot (\frac{1}{M} \sum_{j=1}^{M} w(x_j)) \tag{3.4}$$

To compute an N-sample RIS (Reservoir Importance Sampling) estimator, the RIS algorithm is repeated multiple times, and the results are averaged. The N-sample RIS estimator is defined by the following equation:

$$\langle L \rangle_{ris}^{N,M} = \frac{1}{N} \sum_{i=1}^{N} (\frac{f(y_i)}{\hat{p}(y_i)} \cdot (\frac{1}{M} \sum_{j=1}^{M} w(x_i j))) \tag{3.5}$$

It is seen that the property of RIS method being free from bias is retained when $M = N \geq 1$ and $p, \hat{p} > 0$ at all points where $f \neq 0$. Although M and N can be chosen without restriction, an optimal rate of M/N exists, which depends on variance and relative cost of $\hat{p}$ and f. However, they may face some problems in practice in deciding this rate beforehand. Therefore, the optimum constant of candidate samples, M, for each of the independent samples y_i is usually estimated. It is noteworthy to mention that N represents the number of datasets and going forward we will assume $N = 1$ keeping the explanation simple. However, if N goes beyond 1, our estimators can be easily expanded with the help of the averaging of N individual runs where at each run, M independent candidate samples are considered.

In more broad terms, each pixel q in the image has its own integrand function f_q, as well as the target probability density function $\hat{p}_q$. To give additional attention to this dependency, as of now onwards we will use subscript notation. Figure 3.1, below, depicts the pseudo-code for RIS The justification for selecting these parameters is discussed in the following sub-sections.

3.3 Weighted reservoir sampling(WRS)

The basic approach of weighted reservoir sampling is to draw an element when a new value is read and then to use reservoir sampling to draw the other $N - 1$ values. The fundamental idea of the algorithm is based on Reservoir sampling with a modification that takes into account the weights of stream elements. In the first, the algorithm sets up an empty reservoir of size N and, in the second, sequentially examines every element in the stream. At each time, the algorithm also decides whether to place the current element in the reservoir in part based on the weight of the element by comparing it to a random threshold. Such large weights make those particular elements to be favoured in the sampling process hence increasing the probability of having a sample that is representative of the distribution towards which the sampling is being done. The weights and biases of the reservoir are preserved by changing it whenever selecting an element and then readjusting the

```
Algorithm 1: Resampled Importance Sampling.
Input: M, q: number of candidates to generate (M > 1) for pixel q.
Output: Sample y and the sum of RIS weights Wsum

1. Initialize sets and variables:
    x = {}
    w = {}
    Wsum = 0

2. Generate M candidate samples and their weights:
    for i = 1 to M do
        generate sample xi ~ q(x)
        x = x ∪ {xi}
        wi = p(xi) / q(xi)
        Wsum = Wsum + wi
        w = w ∪ {wi}

3. Resample from the candidates:
    Compute normalized CDF C from weights w
    draw random index z ∈ [0, M) using C to sample according to w
    y = x[z]

4. Return results:
    return y, Wsum
```

Figure 3.1: Resampled Importance Sampling

remaining elements in the reservoir account. Procedures of this type ensure that the last reservoir contains in fact a random sample from the stream while keeping the relative frequencies of the elements into consideration. In this way, the reservoir can be managed efficiently while keeping the running sum of weights under check to ensure that they do not hamper the working of the computer.

$$P_i = \frac{w(x_i)}{\sum_{j=1}^{M} w(x_j)} \tag{3.6}$$

The resemblance between the weighted reservoir sampling algorithm and the probability formula for selecting a random element from a candidate pool derived from the source PDF in RIS is indeed striking. The pseudo-code for the WRS algorithm can be found in the paper cited as [12].

```
Algorithm 2: Weighted Reservoir Sampling

Class Reservoir:
    y <- 0  // The output sample
    Wsum <- 0  // The sum of weights
    M <- 0  // The number of samples seen so far

    function update(xi, wi):
        Wsum <- Wsum + wi
        M <- M + 1
        if rand() < (wi / Wsum) then
            y <- xi

function reservoirSampling(S):
    r <- new Reservoir()
    for i = 1 to length(S) do:
        r.update(S[i], weight(S[i]))
    return r.y
```

Figure 3.2: Weighted reservoir sampling

In the given context, the most suitable algorithm is the WRS algorithm, which is used by comparing a random number in the interval [0...1] with the weight of the element, adjusted for its relative weight. This relative weight is defined as the relative ratio between the weight of the current element and the sum of the weights of the earlier elements in the weighted structure. Another variable M is also employed to record the number of samples that were observed up to the current point, which will be used in the computation of the last estimator X. If the source PDF and the target PDF are the same, then the current sample is weighed according to the distribution of the source PDF as previously explained.

3.4 Streaming RIS using reservoir sampling

As stated before, the application of the WRS algorithm to RIS for transforming it into a streaming algorithm is a straightforward process. The paper provides a pseudo-code that outlines the steps as follows.

```
Algorithm 3: Streaming RIS Using Weighted Reservoir Sampling

foreach pixel q in Image do:
    shadePixel(RIS(q), q)

function RIS(q):
    r <- new Reservoir()  // Initialize a new Reservoir object
    for i = 1 to M do:
        xi <- generateSample(q)  // Generate a sample xi
        wi <- Pq(xi) / p(xi)  // Compute the importance weight
        r.update(xi, wi)  // Update the reservoir with the sample and its weight
    r.Wsum <- adjustWeight(r.Wsum)  // Apply Equation (6) to adjust the weight
    return r

function shadePixel(r, q):
    return fq(r.y) * r.Wsum  // Return the shaded value for pixel q
```

Figure 3.3: Streaming RIS using reservoir sampling

The algorithm starts by identifying M samples as potential samples. For each of the samples we calculate their respective weights depending on the source, and the target Probability Density Functions. Having computed the sample and its weight, we update the above-mentioned reservoir with the approach given previously. This is accomplished by cycling through all initial candidates, which are a subset of the whole list of lights in the direct lighting context. For instance, it is possible to randomly choose M lights out of total N in the scene for performing the reservoir updates during the rendition.

In this way, it is possible to update the reservoir using fewer candidates from both the original and current databases as needed, thus reducing the computational load without losing the ability to sample representative information about lighting. In conclusion, these techniques allow RIS to be efficiently turned into a streaming algorithm, with outstanding results on the render attribute of light sampling.

3.5 Spatiotemporal Reuse

Reservoirs have a significant advantage - they can be integrated with other reservoirs which in turn means that there is no need to reprocess the input streams as it is necessary in the case of the filters. This property is important because it can simply be the process of merging reservoirs even when some elements of the other stream are not accessible. In the case of merging two reservoirs, we progressively consider the current sample y as a new sample with its weight W_{sum} and insert it into the new reservoir. This makes the calculation of a new reservoir in constant time, thereby, avoiding the iteration of the input elements of the different reservoirs. Only the required reservoirs contain information about their current state because the algorithm is needed to minimize communication between different reservoirs. Here is the pseudo-code for combining reservoirs described by T. Tesfamichael in [12].

```
Algorithm 4: Combining the Streams of Multiple Reservoirs

Input: Reservoirs {r1, r2, ..., rk} to combine
Output: A combined reservoir s equivalent to the concatenated input streams of {r1, r2, ..., rk}

function combineReservoirs(q, r1, r2, ..., rk):
    // Step 1: Initialize a new Reservoir object
    s <- new Reservoir()

    // Step 2: Loop over each reservoir in the input set
    foreach r in {r1, r2, ..., rk} do:
        // Step 3: Extract the sample and compute the adjusted weight
        sample <- r.y
        weight <- Pq(r.y)
        adjusted_weight <- weight * r.Wsum * r.M

        // Update the combined reservoir with the sample and its adjusted weight
        s.update(sample, adjusted_weight)

    // Step 4: Calculate the total number of samples seen by all reservoirs
    s.M <- r1.M + r2.M + ... + rk.M

    // Step 5: Adjust the combined reservoir's weight sum
    s.Wsum <- adjustWeight(Pq(s.y) * s.M * s.Wsum)  // Apply Equation (6)

    // Step 6: Return the combined reservoir
    return s
```

Figure 3.4: Spatiotemporal Reuse

In the outlined algorithm, the process starts with an empty reservoir and requires the reservoir to collect one element in each step of the process. Sizes of arrays are then changed for each reservoir that is to be merged; the new composite reservoir is updated by the current sample of the former reservoir. The weight of the sample is given by the product of the probabilistic weight of the sample, the number of the candidate, and the actual branch PDF at the present sample. This can be used to estimate the level of lighting response corresponding to the current light sample.

Subsequently, the total number of candidates in the current reservoir is obtained by adding the total number of candidates noticed in all the different source reservoirs under consideration for merger. This aggregation enables candidate samples accumulation from various sources, meaning that a new reservoir will provide an overall representation within the candidate reservoir.

Since we have residue reservoirs for both the current pixel and neighboring pixels, it is possible to integrate the two and yield far more positive impacts than using one reservoir for the current pixel. The method assumes that the values of neighboring pixels are sufficiently similar and that the process of using backward differences to estimate the partial derivatives does not introduce significant bias.

While images are commonly constructed as sets of frames and if we are in the area of video games, for the current pixel, we have the opportunity to use the reservoir of the previous frame. In such cases it is possible to simply add the current reservoir to the previous frames reservoir, thereby allowing temporal reuse of the same. This temporal combination can increase the quality of results and conclusions that are drawn from the analysis.

This we can achieve by discarding occluded samples for a selected sample y per

reservoir for each pixel. This occlusion check happen before the spatial and temporal reuse pass, in order to avoid sharing samples that are occluded with neighboring pixels. By making this check in advance, we ensure that neighboring pixels have their reservoirs unaltered from pixels used to calculate them. Below is the pseudo-code of the whole process as outlined in the paper.

```
Algorithm 5: RIS with Spatiotemporal Reuse

Input: Image-sized buffer containing the previous frame's reservoirs (prevFrameReservoirs)
Output: The current frame's reservoirs

function reservoirReuse(prevFrameReservoirs):
    // Step 1: Initialize the reservoirs array
    reservoirs <- new Array[ImageSize]

    // Step 2: Generate initial candidates
    foreach pixel q in Image do:
        reservoirs[q] <- RIS(q)  // Use Algorithm 3 to generate initial candidates

    // Step 3: Evaluate visibility for initial candidates
    foreach pixel q in Image do:
        if shadowed(reservoirs[q].y) then
            reservoirs[q].Wsum <- 0  // Set weight to 0 if the sample is shadowed

    // Step 4: Temporal reuse
    foreach pixel q in Image do:
        q' <- pickTemporalNeighbor(q)  // Select a temporal neighbor
        reservoirs[q] <- combineReservoirs(q, reservoirs[q], prevFrameReservoirs[q'])  // Use Algorithm 4 to combine

    // Step 5: Spatial reuse
    for iteration i = 1 to n do:
        foreach pixel q in Image do:
            Q <- pickSpatialNeighbors(q)  // Select spatial neighbors
            R <- {reservoirs[q'] | q' in Q}  // Collect reservoirs from spatial neighbors
            reservoirs[q] <- combineReservoirs(q, reservoirs[q], R)  // Use Algorithm 4 to combine

    // Step 6: Compute pixel color
    foreach pixel q in Image do:
        Image[q] <- shadePixel(reservoirs[q], q)  // Use Algorithm 3 to compute the pixel color

    // Step 7: Return the current frame's reservoirs
    return reservoirs
```

Figure 3.5: ReSTIR

4

Parameters

4.1 Render Parameters

4.1.1 Resolution

This means that the time taken to produce rendering of a scene directly depends with the resolution the user sets. Specifications like 4K or 8K encompass many pixels and thus need significantly more processing power than current games for the creation of each frame. Although not as CPU-intensive as the first test, it is still explosive on our device with an RTX 3070 GPU, even at 2k. Therefore, as the primary adjusting factor of resolution, we mainly change it from 1280 * 720 to 1920 * 1080. Consequently, lower resolution rates can display content even quicker because the program has fewer pixels to analyze. Overall our perception from observation and records suggest that fps is inversely related to resolution, some of which are described by the below figures. (Figure 4.1)

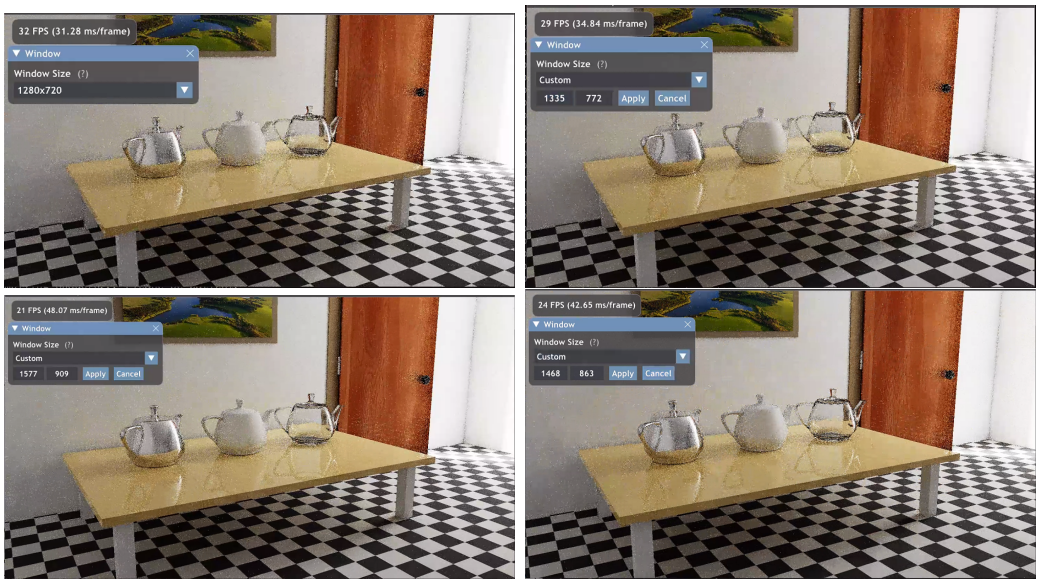
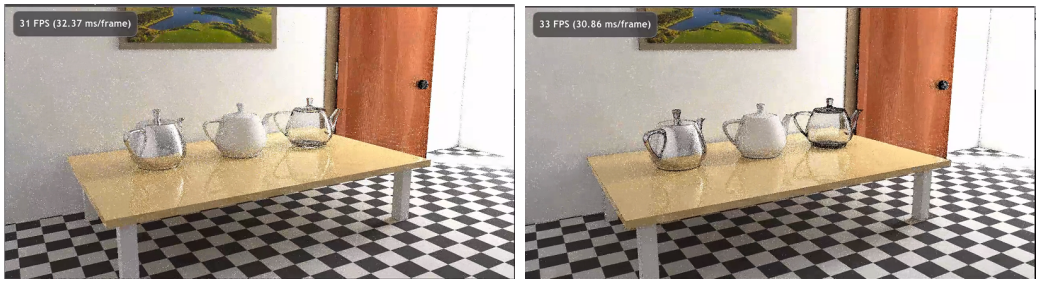


Figure 4.1: fps under various resolutions

4.1.2 Sample Pattern

Sample pattern is for anti-aliasing over multiple frames. The camera jitter is set at the start of each frame based on the chosen pattern. All render passes should see the same jitter. Center disables anti-aliasing by always sampling at the center of the pixel. There are also some other options such as the DirectX sample pattern. The difference is shown in figure 4.2.



(a) Center (b) DirectX

Figure 4.2: Sample Patterns

4.2 ReSTIR Parameters

4.2.1 Candidate Samples

The number of candidate samples can be changed to affect render speed. The more samples are chosen, the slower the render speed will be.

4.2.2 Shift Mapping Modes

The process of Shift Mapping is a very crucial one which was propounded by Lin et al.[14]. For Shift Mapping they employ two types of algorithms although the specific difference between them has not been clearly identified. The reconnection shift [22] sets $y_2 = x_2$ to always reconnect at the first indirect vertex. This can work reasonably well for what are predominantly scenes with diffused lighting. ReSTIR GI [13] implicitly uses this choice but in return, a few heuristics break correctness for performance. A combination of random connection and reconnection [23] that puts off the reconnection through a random connection in the event that connect ability conditions are not met. A variant of this shift mapping is further refined in ReSTIR PT[14].

Reconnection. [22] Reconnection mode seems to go up the fps rate but the shadow and the reflection of the whole picture are wrong. As is shown in the figure, it is evident that the right teapot is fully black, but should have scene resources and therefore should have been transparent. (Figure 4.3)

RandomReplay. In random replay mode, there is a reduction in speed as compared to other modes. While this technique is applied, the three-dimensional object casts

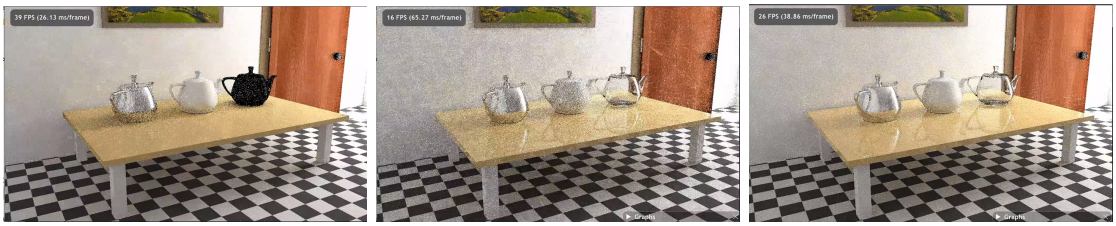


Figure 4.3: From left to right: Reconnection, Random Replay, Hybrid

the right shadow and the transparency effect is good, but it generates more noise compared to reconnection.

Hybrid. The Hybrid mode only allows the user to have the contacting mode on at the same time as the requesting mode not in use. Random Replay mode is useful for checking and monitoring movie video quality since it properly displays shadow and reflection in addition to generating less noise than Random Replay mode. Its fps value can play between reconnection mode and Random replay mode.

4.2.3 Spatial Reuse

Spatial Reuse is introduced in section 3.5. When ReSTIR applies Spatial Reuse, each pixel selects numerous(the number is called Spatial Neighbor Count) random spatial neighbors in a range determined by Spatial Reuse Radius and selected by the resampling algorithm. This step may be executed multiple times.

4.2.3.1 Spatial Neighbor Count

The value influences render speed and denoising effect. Due to figure 4.4, it is obvious that when Spatial Neighbor Count increases from 0 to 2, the denoising works significantly better. However, this improvement is not noticeable anymore when Spatial Neighbor Count grows more than 2. In the meantime, render speeds keep decreasing when we increase the count.

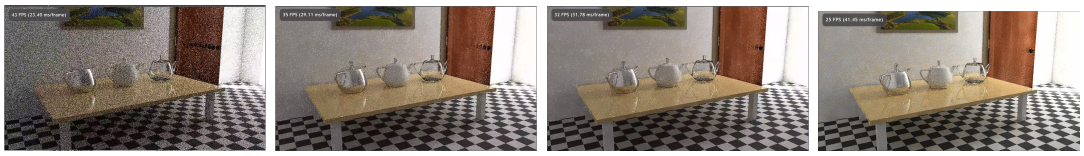


Figure 4.4: From left to right: Spatial Neighbor Count = 0, 1, 2, 6

4.2.3.2 Spatial Reuse Radius

This parameter, Spatial Reuse Radius, denotes how large the range is within which ReSTIR will select the neighbor samples. For example, if the Spatial Reuse Radius $R = 10$, ReSTIR random neighbor selection is within a circle of 10 pixels around a particular point. Due to the observation of figure 4.5, it is not clear whether the performance is better when the R value is large or small. Still, it is evident that the convergence errors reduce when R is larger and in this particular when $R < 20$.

Only when R is greater than 20, a number of noise aspects emerge as the problem. At one point, the noise effect also reaches its highest value when $R = 100$. During the analysis, it was seen that there are no significant differences between different values of Radius in general when it comes to fps.

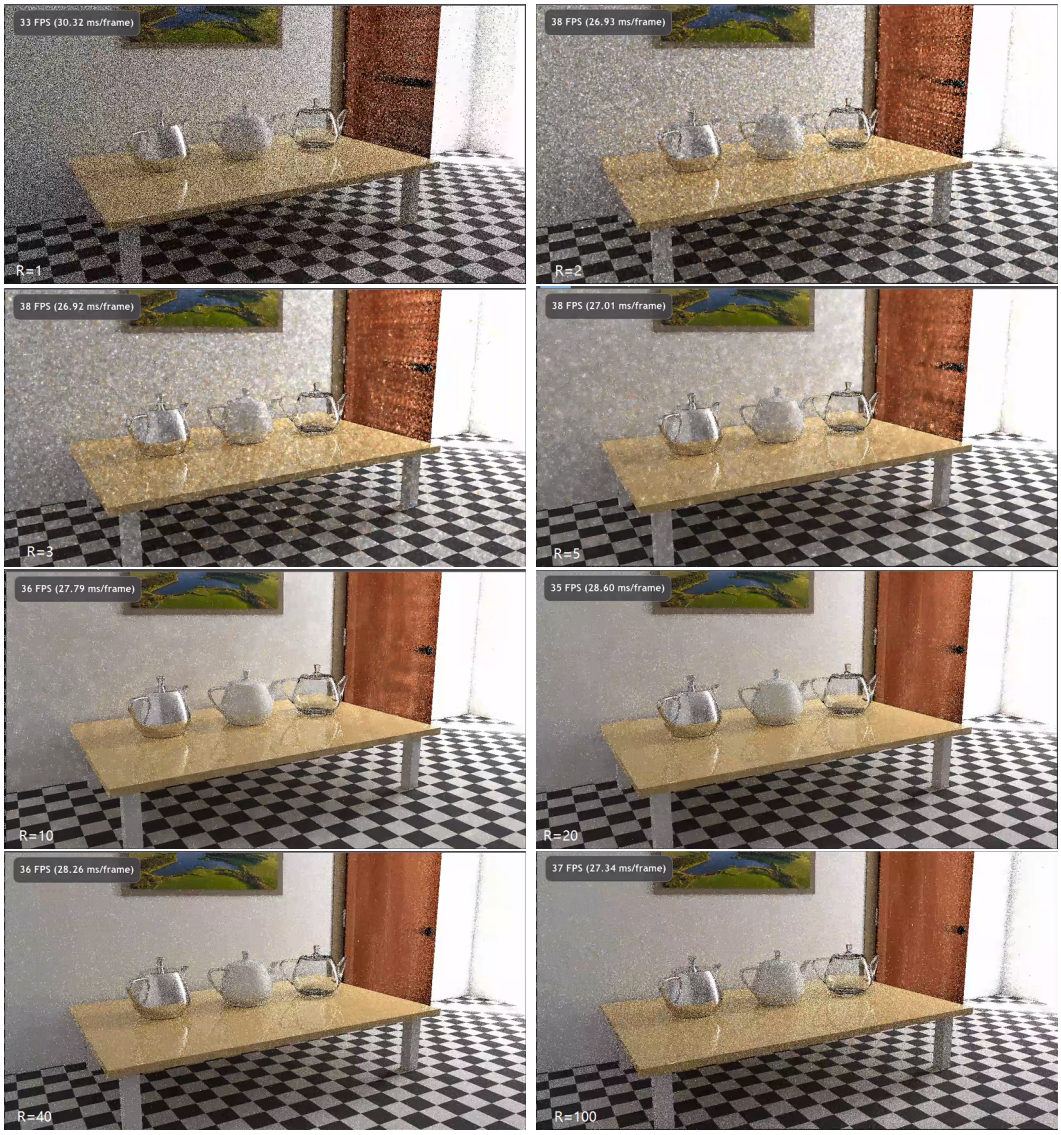


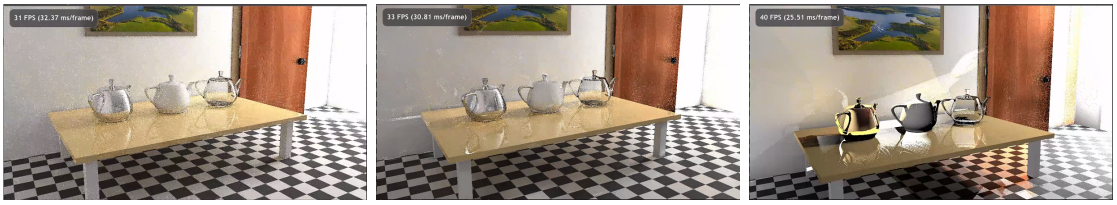
Figure 4.5: Spatial Reuse Radius

4.2.4 Temporal Reuse

4.2.4.1 M-Capping.

In the case of ReSTIR, the last parameter is the number of samples counted as M. However, since M is originally unconstrained in the old ReSTIR algorithm, the relative weights of new samples reduce to zero exponentially as the iteration continues and thus eventually leads to the wrong conclusion. By utilizing M-capping, we want to make M constant as M_c and the relative weight of the temporally reused sample is at most $M_c/M_c + 1$ and this should presumably be good enough to meet

the convergence constraints. A comparison of vibration intensity with M-capping and without M-capping is presented in Figure 4.6. It is noticeable that without M-capping, the render speed becomes higher and higher, causing more and more errors in rendering results.

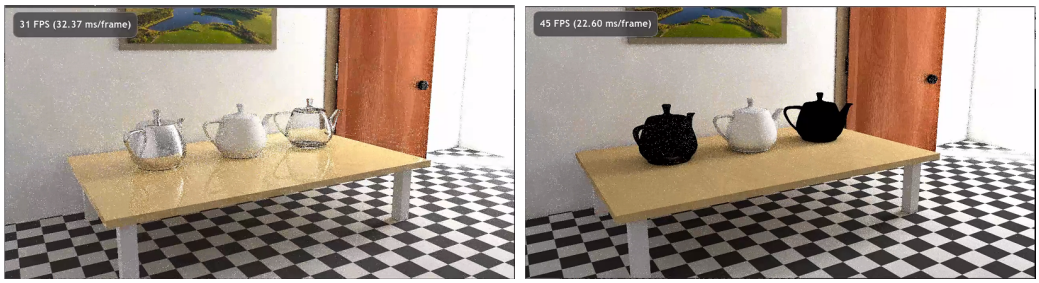


(a) $M - cap : M_c = 20$ (b) No M cap for 30 sec (c) No M cap for 10 min

Figure 4.6: M-capping

4.2.5 Path Sampler Options

BSDF importance sampling. BSDF (Bidirectional Scattering Distribution Function) importance sampling is a technique used in computer graphics and rendering to efficiently sample the incoming and outgoing light directions when calculating the reflection or transmission of light at a surface. When BSDF importance sampling is disabled, the fps will be increased, but all reflections on the glossy surface will vanish. (Figure 4.7)



(a) BSDF Enabled (b) BSDF Disabled

Figure 4.7: BSDF importance sampling

4.3 ToneMapping

Tone mapping can be described as a key approach generally used in image processing and computer graphics to adjust the colors of high-dynamic-range (HDR) images to a level more appropriate for regular devices that have a limited HDR. Due to the nature of Tone mapping is applied with the goal of mapping HDR images with a realistic appearance within the limited dynamic range of the output device.[36]

For the tone mapping parameter used in ReSTIR, the tone mapping operator is used. An example of this is a tone mapping operator that frequently operates on a luminance value or an individual color channel and yields an output value within the

range of [0, 1] anticipated by the display device. Basically, tone mapping operators aim at creating an image that would look as close as possible to what the human eye would where seeing and thus, optimally, transmit the desired tonal mapping without losing credibility.

Using ReSTIR, which is an appropriate tone mapping operator, the HDR colors are transformed to a range possible to display by a viewer resulting in better and realistic scene rendering of the image.

Linear linear tone mapping is a simple direct way of transforming a pixel value of an HDR image to that of the LDR. This type of pixel enhancement focuses on scaling the intensity values linearly so that the overall illumination and contrast remain constant.

By making a linear transformation of the pixel values, which means stretching or shrinking them, all the relative changes in brightness are kept in the same proportion. However, this simplistic approach may lead to some drawbacks, The fact is that this is also a very stringent average, which can give some simplification. In high-contrast scenes, where there are values ranging from deep shadows to bright highlights, the use of linear tone mapping may lead to tone loss and low-contrast images that do not represent the source material well.

However, the image contrasts may appear poor, especially when the darker regions appear excessively darker and shadows may completely hide a scene. Similarly in the dark areas, they look too dark, and thus the details in the dark areas disappear from scene. The loss of details in both dark and bright regions is useful but in some instances compromises the faithfulness of the tone-mapped image to the original scene.

Reinhard Tone Mapping algorithm is one of the most common global tone mapping approaches that appeared in 2002 The Reinhard Tone Mapping algorithm is one of the most common global tone mapping approaches that appeared in 2002 [37]. The suggested techniques purpose is to take a HDR image and create a natural-looking low dynamic range (LDR) rendering of it, without losing the overall histogram range and finer details. This is done using local contrast which is reached with the help of an algorithm that compresses the HDR values while producing an LDR image. Reinhard Tone Mapping has recently emerged as a technique with which artists can create pleasing images that represent a good balance of tonal data that belongs to an HDR scene.

Modified Reinhard The modified version of the Reinhard Tone Mapping algorithm adds one more parameter called the saturation factor, which is described as an important parameter for the tone mapping process that helps to increase the saturation of the picture. The modified algorithm, thus, delivers fine and impressive results with enhanced color intensity based on the adjusted saturation factor. The second control relates, in a way, to the first one and it is used to make fine adjustments to the overall tone mapping process to get the exact amount of color that is wanted in the final image. The expansion of this saturation factor increases the effectiveness and feasibility of the algorithm the designs the HDR scenes in much more aesthetic

and vibrant ways.

ACES The Academy Color Encoding System (ACES) is an open-source, full-range and, three-dimensional, color image encoding system designed by the Academy of Motion Picture Arts and Sciences. It was developed to define a colour-precise workflow and also to introduce the idea that there is no significant difference in the quality of the motion picture images irrespective of their source. ACES allows color to be controlled and passed through from production into post and back again in a structured and safe way, ensuring that color looks the same on a TV, a monitor, a phone screen, a projector or a projector screen. Hence through ACES, filmmakers and other movie production technicians can preserve the original content that they want to portray and rectify or readjust the perception of colors of the intended scene, thus filming or reproducing movies that depict colors accurately as per the intended production. [38]

Accordingly, all the four tone mapping operators as described above are supported by our scene. They do not affect the render speed when toggling between every other one, but toggling between the two will cause a change in contrast, which can be picked up visually by the human eye. (Figure 4.8)

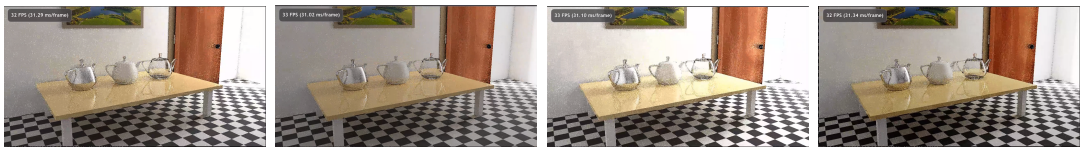


Figure 4.8: From left to right: Linear, Reinhard, Modified Reinhard, ACES

4.4 Scenes

It is important to note that, render speed needed to complete a scene can be different from another scene even when rendering algorithm is the same owing to one reason or the other. Due to this, it was noticed that scenes with varying difficulties can have drastic effects on framerate. Finally, complex scenes with a complicated geometry, a large number of objects and their intricate textures, multiple sources of light, etc., call for more computation and more time to calculate the proper pixel color accurately. Conversely to that, scenes that do not contain as many facets and elements with detailed Op-Amps require less time.

However, if there are more intricate lighting solutions incorporated into a scene, for example, global illumination or ray tracing abilities, rendering time will be significantly higher. Calculating reflections to object surfaces and estimating for shadows on objects as well as, on the light sources all need more mechanisms to compute and, hence, consumes more time. In general terms, whenever there are complicated lighting schemes, it will take more time for the picture to render compared to simpler lighting.

Action sequences, compound objects, complicated simulations like smoke, fire, or fluid and particle systems all take more calculation and time on the renderer. These

effects can perform complex arithmetic and simulation which in return may delay rendering time.

As expected, different ReSTIR scenes were tested. The render speed is truly fluctuating, and what is illustrated in the following picture will explain it.(Figure 4.9) Sharp also gets graphical information and it can be seen that camera angle can alter the render speed even within the same scene, which is linked to the number of triangles that get rendered in the view of the camera.

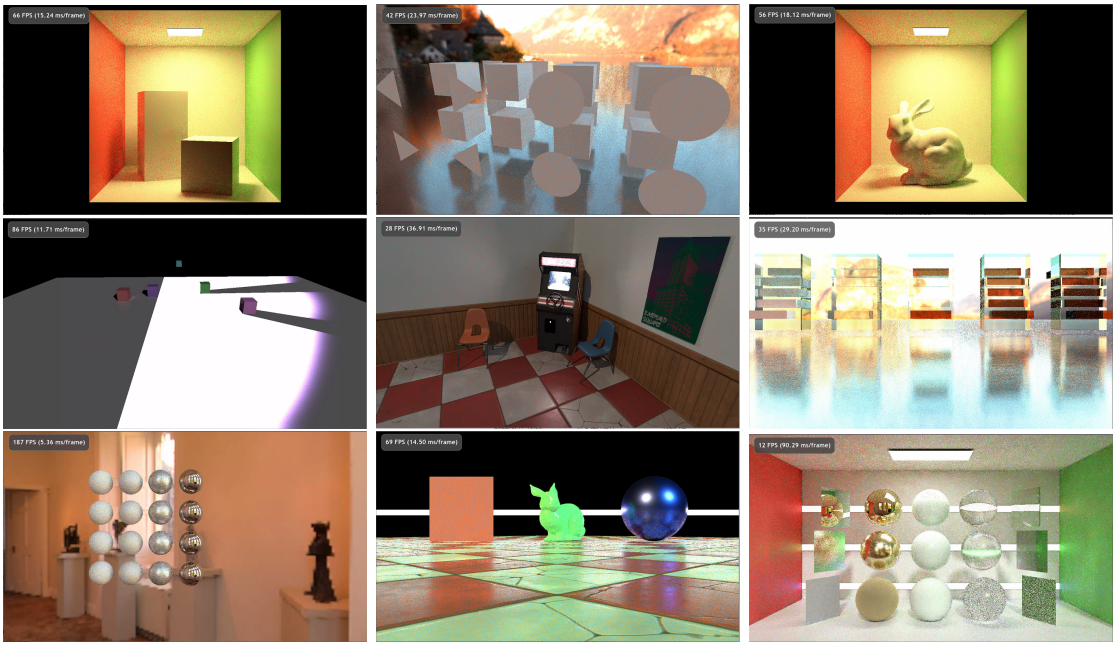


Figure 4.9: Different Scenes

5

Results

5.1 Spatial Reuse

5.1.1 Spatial Neighbor Count

With the experiment in section 4.2.3.1, a plot about how spatial neighbor count affects fps and noise is given below. The fps values is the actual value, while the noise value is standardized in a range of [0,1]. From figure 5.1 we can conclude

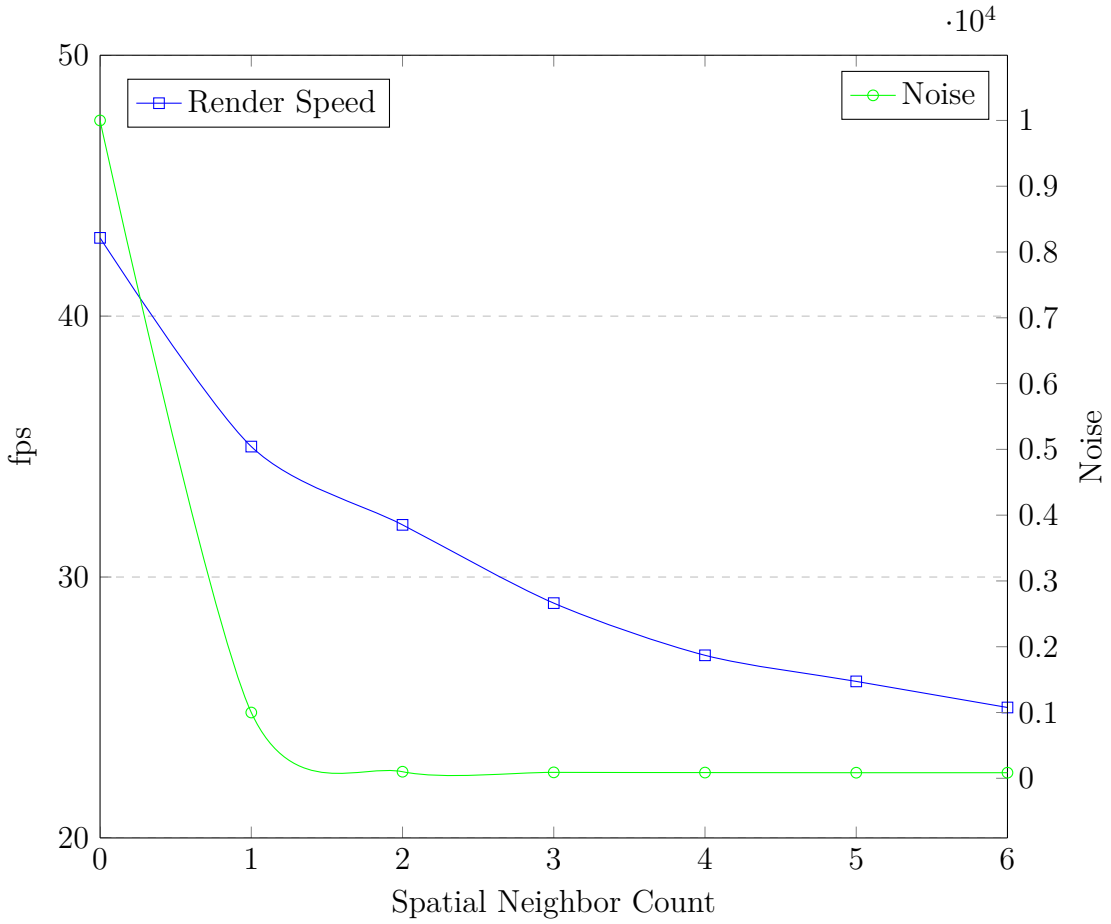


Figure 5.1: Render effect of Spatial Neighbor Count

that fps is continually influenced when the spatial neighbor count increases, but the

denoising works insignificantly when spatial neighbor count is more than 2.

5.1.2 Spatial Reuse Radius

From the figure With the experiment in section 4.2.3.1, another plot about how spatial Reuse Radius affects convergence error and noise is given below. They are valued in a standard of variance.

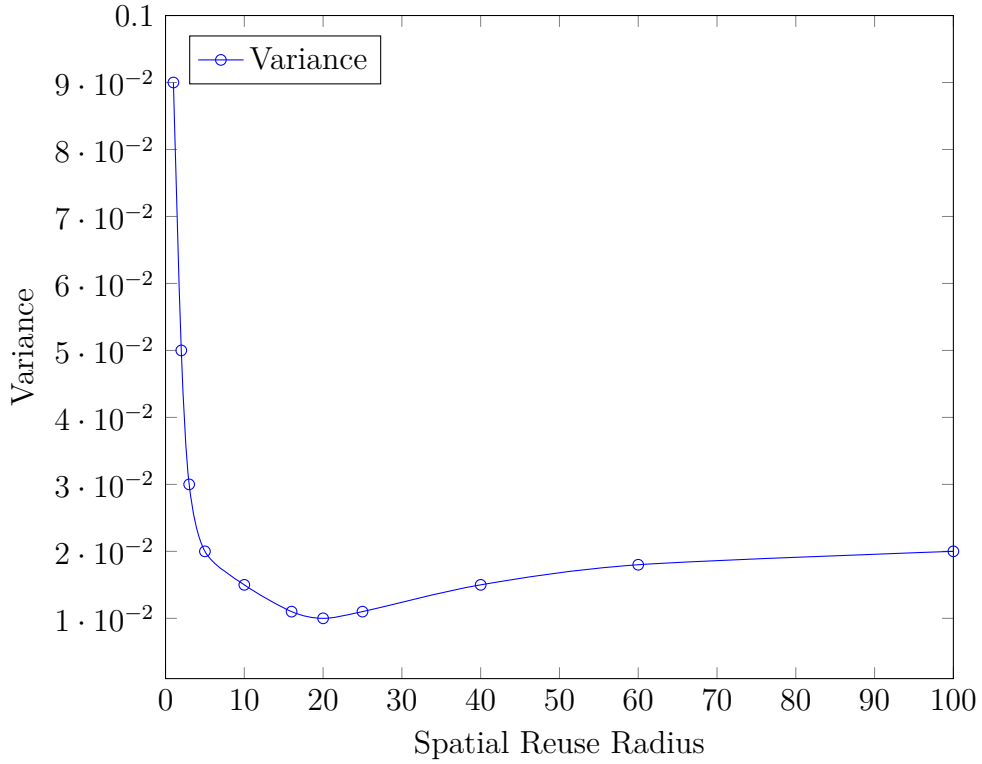


Figure 5.2: Render effect of Spatial Neighbor Count

From the figure 5.2 we can conclude that the variance is at the lowest point when the radius is in the range [16,25]. Even though higher radius produce not much more noise, it is also crucial to find a range that best suits for sampling. And shorter sampling radius should definitely be avoided according to the plot.

5.2 M-capping

How M-capping effects rendering speed and convergence to a wrong result due to temporal correlations is partly shown in 4.2.4. But there is still not enough discussion on the effects among different M cap values. Thus we do the statistics over ReSTIR PTs integration error with temporal reuse and increasing frame counts. Colors correspond to different M-cap values; the scene is static to avoid errors from animation.

Pixels compute a new independent sample on each iteration, which is resampled with the temporal result from the prior frame.

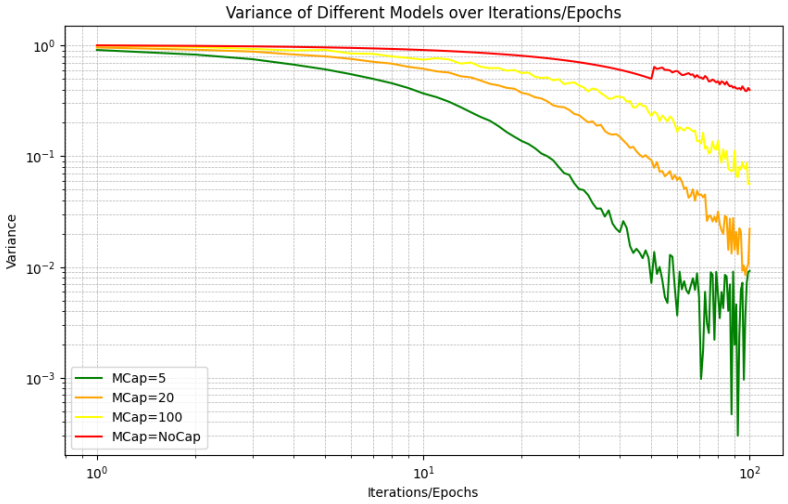


Figure 5.3: Effect of M-cap

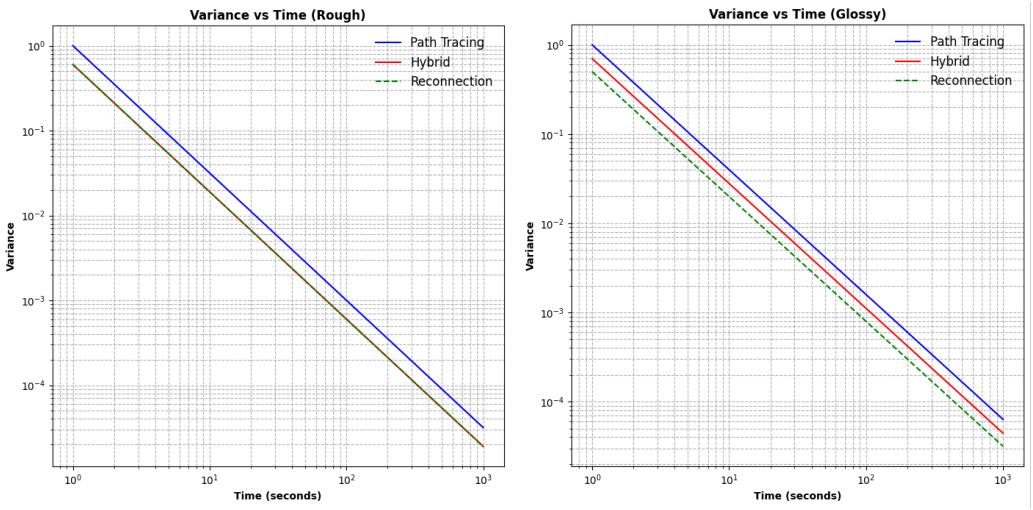
From the figure 5.3 we know that the error of ReSTIR PT with temporal reuse varies with increasing frame counts and different M-cap values. A large M-cap eventually increases noise, while low values do not minimize error. Good M-caps (green) should give consistently low errors. It also implies that M-cap value should be carefully chosen, to causing less error and make the noise minimum.

5.3 Shift Mapping

A discussion among different shift mapping modes has already been given in 4.2.2. From that section, it is roughly acknowledged that reconnection mode works poorly on glossy surfaces, as well as the reflection effects related. But on the other hand, if we apply reconnection mode to rough surfaces, it causes no more error than other modes but has a higher render speed. A plot about how rough and glossy surface influence shift mapping is given with a path tracing baseline as well.

From the figure 5.4 we can see Both shift mappings are good for rough objects, but the hybrid shift (green) is more suited for scenes with glossy surfaces. The reason reconnection shift is efficient for rough surfaces is it allows cheap path reuse from indirect light. Hence the reconnection shift (green) always reconnects after a primary hit, regardless of BSDF, it less efficiently reuses paths involving glossy interactions.[14]

In contrast, hybrid works better with glossy surfaces. It combines Reconnection and Random replay, which is better for handling glossy surfaces. It also keeps the efficiency of Reconnection while dealing with rough surfaces.



(a) Rough (b) Glossy

Figure 5.4: Shift Mapping

6

Conclusion

6.1 Discussion

6.1.1 Ethical Considerations

When it comes to visuals with the improvement of video graphic processing the distinction between the realism of conditions and for example a game is even closer. This could pose negative implications on matters hereby lasting for long before computer screens such as the display of excessive game-playing that can be unhealthy. When young people spend many hours before a computer, it is bad for people's eyes and bodies, because there is a possibility of blood clots when standing up for a long time.

While Virtual Reality has realistic lighting and better graphics, people could find themselves disconnecting from reality and spending most of their time in a virtual environment instead. On the other hand, the immersion that comes with more realistic graphics in games can lead to them being more enjoyable to play, as long as the playing time is kept at reasonable levels.

6.2 Conclusion

We studied how parameters in the ReSTIR algorithm influence the render speed and quality. We introduced the background and related work, which provides the theoretical background of how parameter works in the ReSTIR algorithm. The values of several parameters are adjusted, in order to exploring any possibility of optimizing render speed and denosing quality. We also plot the statistics based on our Engine data and observation.

Given all the mentioned chapters, we can conclude that the render effect of ReSTIR algorithm is influenced by many parameters. Resolution can significantly influence the render speed, but this conclusion applies to a broader aspect of rendering algorithms, not only for ReSTIR. Sample patterns also affect render speed and quality in the subtleties.

When it comes to Spatial Reuse, Spatial Neighbor Count and Spatial Reuse Radius are two important parameters that should be discussed. From section 4.2.3.1, 4.2.3.2 and 5.1, it is evident that the former one influences FPS no matter how high its

value is, but denosing significantly when the value grows from 0 to 2. The latter one has little effect on FPS, but fixing convergence error increasingly when its value grows from 0 to 20, and increases noise slightly when it grows from 20 to 100. So the suggestion on a proper R should be approximately be 20.

M-capping is introduced in ReSTIR PT and helps keeping convergence error to be fixed when rendering time past. From section 4.2.4 and 5.2, we can conclude that Capping M to maximize the benefits of temporal reuse is key to real-time rendering with ReSTIR. And the suggestion on a proper M should be approximately be 20.

Shift mapping Hybrid mode is another parameter adjustment method introduced in ReSTIR PT and has huge influence on rendering as well. From section 4.2.2 and 5.3, we know that previous Reconnection mode is more suitable for roughness scenes, but works less correctly on glossy surface. Random Replay works much better on glossy surfsace but has less efficiency. Hybrid mode perfectly combines both advantages of these two shift mapping, optimizes the render quality on glossy surface of Reconnection, and keeps its efficiency on rough circumstances.

There are also several parameters that can influence real-time rendering with ReSTIR. Different tone-mappings lead to different appearances of color and contrast. And given the different complexities, the render speed is highly various among different scenes. Though these are also factors that only only influences ReSTIR algorithm, it shows the commonality of different rendering mode.

Based on all these results and observations, we can conclude that we have explored many parameters to optimize the rendering of ReSTIR, and give the optimal value of how they should be set to achieve the goal.

Bibliography

[1] I. E. Sutherland, "Sketchpad: A man-machine graphical communication system," in *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*, ser. AFIPS '63 (Spring), Detroit, Michigan: Association for Computing Machinery, 1963, pp. 329–346, ISBN: 9781450378802. DOI: 10.1145/1461551.1461591. [Online]. Available: https://doi.org/10.1145/1461551.1461591.

[2] A. Appel, "Some techniques for shading machine renderings of solids," in *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, ser. AFIPS '68 (Spring), Atlantic City, New Jersey: Association for Computing Machinery, 1968, pp. 37–45, ISBN: 9781450378970. DOI: 10.1145/1468075.1468082. [Online]. Available: https://doi.org/10.1145/1468075.1468082.

[3] I. O. for Standardization, *Information Processing Systems, Computer Graphics, Graphical Kernel System (GKS) Functional Description*. International Organization for Standardization, 1985.

[4] W. Hewitt, "PHIGS - Programmers Hierarchical Interactive Graphics System," *Computer Graphics Forum*, 1984, ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.1984.tb00164.x.

[5] W. StraSSer, "Schnelle kurven-und flächendarstellung auf grafischen sichtgeräten," Ph.D. dissertation, 1974.

[6] T.-P. Bui, "Illumination of computer generated images," *Computer Science, University of Utah, UTEC-CSc-73-129*, pp. 18–6, 1973.

[7] T. Whitted, "An improved illumination model for shaded display," in *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, 1979, p. 14.

[8] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86, New York, NY, USA: Association for Computing Machinery, 1986, pp. 143–150, ISBN: 0897911962. DOI: 10.1145/15922.15902. [Online]. Available: https://doi.org/10.1145/15922.15902.

[9] H. W. Jensen, "Global illumination using photon maps," in *Rendering Techniques '96*, X. Pueyo and P. Schröder, Eds., Vienna: Springer Vienna, 1996, pp. 21–30, ISBN: 978-3-7091-7484-5.

[10] E. Veach and L. Guibas, "Bidirectional estimators for light transport," in *Photorealistic Rendering Techniques*, G. Sakas, S. Müller, and P. Shirley, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 145–167, ISBN: 978-3-642-87825-1.

[11] J. F. Talbot, "Importance resampling for global illumination," English, Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-03-07, Ph.D. dissertation, 2005, p. 89, ISBN: 9798662551510. [Online]. Available: http://proxy.lib.chalmers.se/login?url=https://www.proquest.com/dissertations-theses/importance-resampling-global-illumination/docview/2452104462/se-2.

[12] B. Bitterli, C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz, "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020, ISSN: 0730-0301. DOI: 10.1145/3386569.3392481. [Online]. Available: https://doi.org/10.1145/3386569.3392481.

[13] Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni, "Restir gi: Path resampling for real-time path tracing," *Computer Graphics Forum*, vol. 40, no. 8, pp. 17–29, 2021. DOI: https://doi.org/10.1111/cgf.14378. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14378. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14378.

[14] D. Lin, M. Kettunen, B. Bitterli, J. Pantaleoni, C. Yuksel, and C. Wyman, "Generalized resampled importance sampling: Foundations of restir," *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022, ISSN: 0730-0301. DOI: 10.1145/3528223.3530158. [Online]. Available: https://doi.org/10.1145/3528223.3530158.

[15] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, 1977, pp. 192–198.

[16] E. Veach and L. Guibas, "Bidirectional estimators for light transport," in *Photorealistic Rendering Techniques*, Springer, 1995, pp. 145–167.

[17] E. Veach and L. J. Guibas, "Optimally combining sampling techniques for monte carlo rendering," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 419–428.

[18] H. W. Jensen, "Global illumination using photon maps," in *Rendering Techniques 96: Proceedings of the Eurographics Workshop in Porto, Portugal, June 17–19, 1996 7*, Springer, 1996, pp. 21–30.

[19] P. Bekaert, M. Sbert, and J. H. Halton, "Accelerating path tracing by re-using paths.," *Rendering Techniques*, vol. 2, pp. 125–134, 2002.

[20] P. Bauszat, V. Petitjean, and E. Eisemann, "Gradient-domain path reusing," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–9, 2017.

[21] M. Kettunen, M. Manzi, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker, "Gradient-domain path tracing," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–13, 2015.

[22] J. Lehtinen, T. Karras, S. Laine, M. Aittala, F. Durand, and T. Aila, "Gradient-domain metropolis light transport," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–12, 2013.

[23] B.-S. Hua, A. Gruson, V. Petitjean, *et al.*, "A survey on gradient-domain rendering," in *Computer Graphics Forum*, Wiley Online Library, vol. 38, 2019, pp. 455–472.

[24] M. Manzi, M. Kettunen, F. Durand, M. Zwicker, and J. Lehtinen, "Temporal gradient-domain path tracing," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–9, 2016.

[25] M. Manzi, M. Kettunen, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker, "Gradient-domain bidirectional path tracing.," in *EGSR (EI&I)*, 2015, pp. 65–74.

[26] A. Gruson, B.-S. Hua, N. Vibert, D. Nowrouzezahrai, and T. Hachisuka, "Gradient-domain volumetric photon density estimation," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[27] B.-S. Hua, A. Gruson, D. Nowrouzezahrai, and T. Hachisuka, "Gradient-domain photon density estimation," in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 31–38.

[28] W. Sun, X. Sun, N. A. Carr, D. Nowrouzezahrai, and R. Ramamoorthi, "Gradient-domain vertex connection and merging.," in *EGSR (EI&I)*, 2017, pp. 83–92.

[29] V. Petitjean, P. Bauszat, and E. Eisemann, "Spectral gradient sampling for path tracing," in *Computer Graphics Forum*, Wiley Online Library, vol. 37, 2018, pp. 45–53.

[30] J. Van de Woestijne, R. Frederickx, N. Billen, and P. Dutré, "Temporal coherence for metropolis light transport," in *Eurographics Symposium on Rendering-Experimental Ideas & Implementations*, Eurographics Association, 2017, pp. 55–63.

[31] E. Veach and L. J. Guibas, "Metropolis light transport," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 65–76.

[32] L. Tessari, J. Hanika, and C. Dachsbacher, "Local quasi-monte carlo exploration.," in *EGSR (EI&I)*, 2017, pp. 71–81.

[33] G. Boissé, "World-space spatiotemporal reservoir reuse for ray-traced global illumination," in *SIGGRAPH Asia 2021 Technical Communications*, 2021, pp. 1–4.

[34] X. Hermans, "The effectiveness of the restir technique when ray tracing a voxel world," M.S. thesis, 2022.

[35] R. Sawhney, D. Lin, M. Kettunen, *et al.*, "Decorrelating restir samplers via mcmc mutations," *arXiv e-prints*, arXiv–2211, 2022.

[36] Wikipedia contributors, *Tone mapping — Wikipedia, the free encyclopedia*, `https://en.wikipedia.org/w/index.php?title=Tone_mapping&oldid=1131211403`, [Online; accessed 5-June-2023], 2023.

[37] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '02, San Antonio, Texas: Association for Computing Machinery, 2002, pp. 267–276, ISBN: 1581135211. DOI: `10.1145/566570.566575`. [Online]. Available: `https://doi.org/10.1145/566570.566575`.

[38] Wikipedia contributors, *Academy color encoding system — Wikipedia, the free encyclopedia*, [Online; accessed 6-June-2023], 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Academy_Color_Encoding_System&oldid=1145722448.