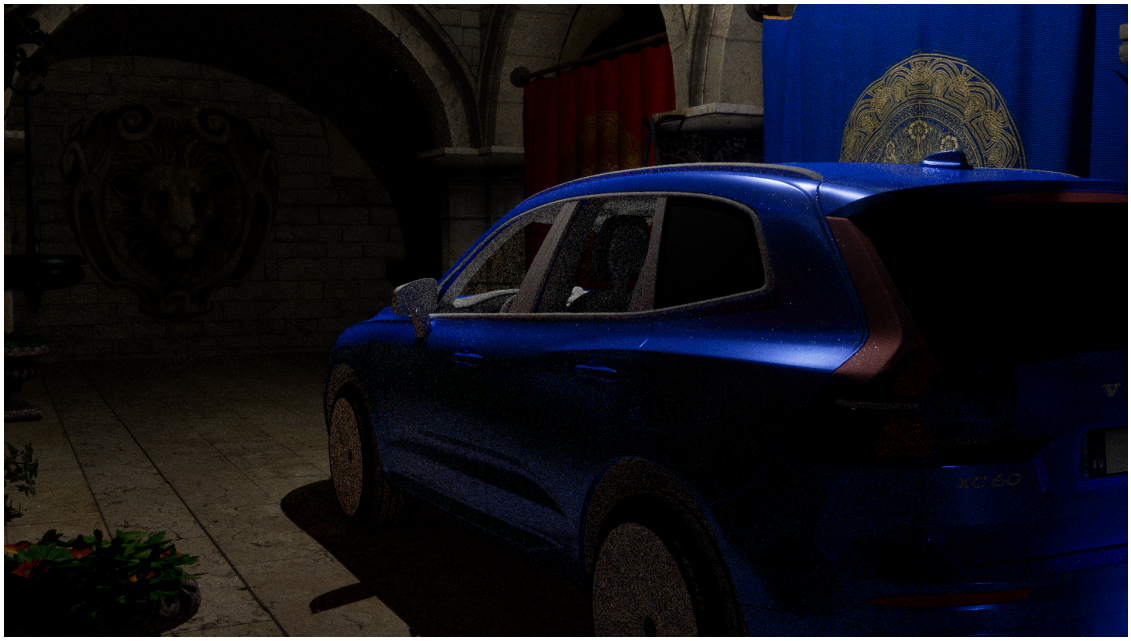


Optimizing Night Driving Simulations: A Comparative Study of Light Simulation Software

Master's thesis in Computer science and engineering

Lanyu Liang

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Optimizing Night Driving Simulations: A Comparative Study of Light Simulation Software

Lanyu Liang

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Optimizing Night Driving Simulations: A Comparative Study of Light Simulation Software
Lanyu Liang

Supervisor: Ulf Assarsson, Department of Computer Science and Engineering
Advisor: Henrik Mattsson, Volvo Cars
Examiner: Ulf Assarsson, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A volvo XC60 in Sponza at night. Render in Unity

Typeset in LaTeX
Gothenburg, Sweden 2025

Optimizing Night Driving Simulations: A Comparative Study of Light Simulation Software

Lanyu Liang
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Night driving simulations are essential tools for the development and validation of advanced automotive lighting systems. These simulations require not only photometrically accurate representations of light-material interactions but also real-time performance suitable for iterative design and driver-in-the-loop evaluations. This thesis presents a comparative study of three rendering platforms: Ansys AVxcelerate Headlamp, Synopsys LucidDrive, and Unity, to assess their capabilities in simulating nighttime driving environments with high visual fidelity.

The study benchmarks these tools across several dimensions, including rendering performance, photometric accuracy, and perceptual similarity, using standardized test scenes and real-world photometric profiles. A particular focus is placed on evaluating Unitys real-time ray tracing capabilities, enhanced by ReSTIR, against the more static and proprietary pipelines of AVxcelerate and LucidDrive. Experiments utilize standardized test scenes and automotive-grade models under controlled nighttime conditions, profiling performance via structural similarity of rendering results, GPU utilization, frame rate, and memory consumption.

The results underscore the limitations of commercial tools in handling dynamic lighting scenarios and complex BRDFs, while highlighting the flexibility and performance of open rendering frameworks. This work provides a reproducible benchmarking methodology and lays the foundation for future research on hybrid rendering strategies, perceptual validation models, and real-time simulation of intelligent headlight systems.

Keywords: Computer graphic, ReSTIR, SSIM, Real-time rendering, Global illumination, Photometric validation, Automotive lighting.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my academic supervisor, Professor Ulf Assarsson, for his invaluable guidance, technical insights, and continuous support throughout the course of this thesis.

I would also like to thank my industrial advisor, Henrik Mattsson and Jakob Eriksson, at Volvo Cars, for providing not only access to real-world use cases and simulation tools but also valuable feedback. This connected this research to practical challenges in the automotive industry.

I would like to thank Lucile Cassaing from Ansys for the technical support.

Finally, I would like to thank my family and my friends, for their unwavering support, patience, and motivation throughout my studies.

Lanyu Liang, Gothenburg, 2025-06-18

Contents

List of Figures

List of Tables

This page was intentionally left blank.

1

Introduction

Driving during nighttime conditions presents notable safety challenges, including reduced visibility, glare effects that impair reaction times, and increased fatigue-related risks [1]. These factors contribute disproportionately to collision rates, motivating the development of advanced vehicle safety systems with software-controlled adaptive lighting and driver assistance features. Modern automotive software architectures now enable dynamic control of headlamp patterns and intensity, requiring simulation platforms to validate these technologies due to a trend of continuously shortening development times and increase complexity.

Volvo Cars exemplifies a strong commitment to this field through its innovative work in exterior lighting. As a company known for its longstanding focus on safety and Scandinavian design, Volvo leverages cutting-edge, physics-based simulation tools to evaluate headlamp visibility, homogeneity, and function in virtual environments. The Exterior Lighting department at Volvo is an integral part of a global, cross-functional development team that collaborates closely with design, engineering, and perceived quality departments. The team is responsible for delivering distinctive, high-performance lighting concepts such as the iconic 'Thors Hammer' headlights. Their work not only embodies Volvos aesthetic values but also addresses technical challenges like glare management and light distribution accuracy in low-light conditions. Exterior lighting is also a key communication point between the vehicle and its environment, becoming increasingly vital as new technologies and styling trends advance.

Volvo representatives have reported that current industry solutions used for night driving simulations, such as Ansys AVxcelerate, exhibit limitations in accurately representing light-material interactions. These shortcomings stem from restricted control over rendering engines and insufficient realism in bidirectional reflectance distribution functions (BRDFs), particularly for challenging materials like asphalt and retroreflective signage [1]. This highlights a critical need for more sophisticated and controllable rendering frameworks in automotive lighting simulation. Proprietary tools mention that this is an observation of computational efficiency over physical precision, compromising their ability to simulate glare dynamics from oncoming headlights or spatially varying road illumination. This work benchmarks the performance of established commercial simulators against emerging ray tracing methodologies, quantifying disparities in irradiance accuracy, temporal stability, and angular light distribution fidelity. By correlating simulation outputs with photometric mea-

surement standards, we establish a framework for improving predictive capability in low-light driving scenarios while maintaining real-time performance constraints. Specifically, we evaluate the effectiveness of ray tracing through Unity's scriptable rendering pipeline to assess headlamp performance under nighttime conditions.

As urbanization expands, the demand for high-fidelity night driving simulators has grown increasingly critical. These systems enable controlled practice of hazard recognition and navigation under low-light conditions while avoiding real-world risks. However, their effectiveness depends on accurately replicating complex light-material interactions and photometric distributions inherent to nighttime environments. Existing approaches often struggle to balance computational efficiency with physical accuracy, particularly in simulating artificial light sources (e.g., headlights, streetlights) and material reflectance properties like asphalt and retroreflective signs. This work accomplishes this by evaluating whether next-generation ray-tracing technologies can compensate for these shortcomings while maintaining rigorous photometric validation standards.

1.1 Technical objectives

This study focuses on two interconnected pillars of night driving simulation:

- **Rendering Architecture**: Comparative analysis of pre-rendering versus real-time global illumination strategies, quantifying trade-offs between visual accuracy and computational load.

- **Validation**: Quantitative assessment using Structural Similarity Index Measure (SSIM) to align simulated light distributions with real-world photometric data.

1.2 Innovation and scope

By integrating hardware-accelerated ray tracing [2], this framework aims to surpass current industry standards in both realism and measurement accuracy. The evaluation includes direct comparisons with two commercial software solutions, analyzing their algorithmic approaches to light transport and hardware utilization. Notably, this work focuses specifically on light behavior simulation rather than full vehicle dynamics or traffic modeling, enabling targeted optimization of visual-physical congruence. This thesis outlines progress in system implementation, presents preliminary validation results, and identifies key challenges in maintaining radiometric precision while achieving real-time performance, a critical requirement for automotive lighting evaluation and simulation [3]. Subsequent sections elaborate on the technical methodology, experimental validation protocols, and road map for future enhancements.

1.3 Automotive lighting simulation software

In this thesis, we evaluate and benchmark three prominent simulation platforms widely used in the automotive industry for night driving and lighting system development: Ansys AVxcelerate, Synopsys LucidDrive, and Unity. AVxcelerate seems currently focused on comprehensive sensor simulation and integration with various driving simulators, while Synopsys LucidDrive seems to excel in headlamp beam pattern evaluation and real-time simulation. In addition, Unity will be used to test how ray tracing effect to the night driving simulator. These tools represent a spectrum of capabilities, from highly specialized optical simulation engines to general-purpose real-time rendering frameworks. Each platform offers distinct advantages and trade-offs regarding physical accuracy, rendering fidelity, integration capabilities, and computational performance. By analyzing their approaches to photometric realism and BRDF implementation, we aim to contextualize the current state of the art and highlight areas for innovation.

1.3.1 Ansys AVxcelerate Headlamp

Ansys AVxcelerate Headlamp (2024R2) is a real-time, physically-based optical simulation solution specifically designed for virtual testing and verification of automotive headlamps. It enables optical engineers to simulate and test headlamp designs and control software in a virtual environment without the need for costly and time-consuming physical prototyping and night-time road testing [4].

1.3.2 Synopsis Luciddrive

LucidDrive (2024.03) is a real-time night driving simulation software in the LucidShape series of products from Synopsys. It can be seamlessly integrated with other optical design software, such as LucidShape, to utilize the light intensity distribution data they generate for more accurate simulation [5].

1.3.3 Unity

Unity (6000.0.44f1) is a cross-platform game engine, but its powerful real-time 3D graphics rendering capabilities and ease-of-use have led to its use in a wide range of industries beyond game development, especially in industrial rendering and simulation. It can be used to simulate a car cockpit environment and to test the design and interaction of human machine interfaces such as dashboards and entertainment systems [6].

1.4 Ethical considerations

This thesis focuses on the simulation and analysis of automotive headlamp systems under low-light driving conditions, with the aim of improving safety and visibility through more accurate and physically realistic modeling. While the research does

not involve direct human participants or sensitive data, several ethical considerations must still be acknowledged.

Given the safety-critical nature of headlamp performance, there is an ethical responsibility to ensure that the simulation methods developed or benchmarked in this work are as accurate and transparent as possible. Misleading or overly optimistic results from simulations could influence design decisions that, in real-world use, might compromise road safety. The thesis aims to mitigate this risk by rigorously comparing simulated outputs to photometric standards and highlighting limitations in both commercial and ray-traced approaches.

High-fidelity simulations, especially those employing ray tracing, can be computationally intensive, requiring significant energy. While the immediate environmental impact of a research study might be small, scaling up these simulations for widespread industrial use could have energy consumption implications. Developers should consider optimizing algorithms and leveraging energy-efficient hardware to mitigate this.

2

Related Work

Simulating nighttime driving environments has gained significant attention in both academic and industrial. This interest is predominantly driven by the imperative for robust virtual testing of advanced headlamp systems, driver assistance technologies, and autonomous vehicle sensors [7]. Consequently, a range of studies have proposed methodologies aimed at enhancing the physical accuracy, visual realism, and computational efficiency of such simulations.

Villa *et al.* [1] investigated the application of high dynamic range (HDR) rendering and real-time shading models for visualizing automotive lighting systems in low-light conditions. Their work underscored the importance of perceptual realism, demonstrating the influence of tone mapping and exposure adaptation on human evaluation of lighting performance. A notable limitation, however, was the reliance on proprietary rendering pipelines, which constrained the reproducibility and flexibility of their approach.

Neale *et al.* [8] conducted a space analysis to assess the validity of using standard photometric data filescommonly employed for street lighting and building illuminationin simulating vehicle headlamp light distribution. Their study systematically evaluated discrepancies between simulated light distributions derived from photometric files and actual measurements taken from real headlamps. By comparing light values calculated in a simulated environment to local measurements, they identified significant differences, highlighting the complexities involved in accurately modeling headlamp light patterns. The authors discussed how the light distribution characteristics of photometric files can differ from those of actual headlamp assemblies, emphasizing the need for careful evaluation when incorporating such data into lighting simulations. While their research established a strong foundation for validation methodologies, it did not incorporate modern rendering techniques such as real-time ray tracing or perceptual metrics like the Structural Similarity Index Measure (SSIM).

Keller *et al.* [9] introduced a hybrid simulation framework which represents a significant contribution to the field of physically based rendering. This framework offers an unbiased rendering approach, typically utilizing path tracing or photon mapping to simulate intricate light interactions within a scene. A key aspect of the design is its GPU acceleration on NVIDIA's CUDA architecture. Although this approach greatly improves visual fidelity and performance, its main focus remains on the simulation of scene environments such as vehicle interiors, neglecting the limitations of

light distribution in automotive industry simulations.

More recently, Ouyang *et al.* [10] and other researchers have explored the integration of advanced light transport algorithms, including ReSTIR (spatio-temporal reservoir resampling), into real-time graphics engines. These methodologies have shown promising results in noise reduction and temporal stability, particularly under constrained sample budgets in urban and night-time scenes. However, the applicability of these studies has largely been confined to synthetic benchmarks or indoor scenarios, with their relevance to automotive lighting validation remaining unexplored.

Outlines the previous works, this thesis presents a directly compares of three distinct simulation platforms: Ansys AVxcelerate, Synopsys LucidDrive, and Unity with ReSTIR integration. This evaluation uses a common test scenario, allowing for a rigorous and consistent comparison. Unlike prior research, which often concentrated on a single software tool or broad light transport principles, this study specifically addresses the critical intersection of real-time performance, photometric accuracy, and perceptual quality in the context of simulating night driving environments. This comprehensive approach is targeted to offer a broader and more practical understanding of the inherent trade-offs involved in night-time driving simulations, thereby contributing novel insights to the field.

3

Theory

Realistic light simulation in real-time rendering requires physically-based lighting models, efficient algorithms, and precise photometric unit adaptation. This section elaborates on four key aspects: the rendering equation, light source modeling, photometric units and implementation strategies. Understanding how light behaves and is measured is fundamental for simulating physically plausible lighting, especially when using real-world, measured light sources in rendering.

3.1 Simulating physically-plausible lighting in real-time rendering

3.1.1 Introduction to radiometry and photometry

Radiometry, the science of measuring electromagnetic radiation (including visible light), provides a precise mathematical framework to define quantities such as radiance, irradiance, and radiant intensity. These quantities form the foundation of the rendering equation used in physically based rendering models (see Equation 3.2), and are summarized in Table 3.1.

The branch of optics that deals with the measurement of light is called radiometry. Radiation metrology defines a set of basic physical quantities used to measure optical radiation, and the metrics have become the most important basic concepts in computer graphics and are applied in Equation 3.2, with several commonly used physical quantities and units shown in Table 3.1.Here, *sr* stands for steradians, the SI unit of solid angle. In three-dimensional geometry, a solid angle describes how large an object appears to an observer located at a specific point. It is analogous to the concept of planar angles in two dimensions but extended into 3D space. A solid angle Ω is defined as

$$\Omega = \frac{A \cos \theta}{r^2},$$

where A is the area of the surface, r is the distance from the viewpoint to the surface, and θ is the angle between the viewing direction and the surface normal. For a sphere, the total solid angle around a point is 4π sr, and a region on a unit sphere with area 1 corresponds to 1 sr.

Name	Unit	Symbol
radiant energy	J	Q
radiant flux	W	Φ
irradiance	W/m^2	E
radiant intensity	W/sr	I
radiance	$W/(m^2 \cdot sr)$	L

Table 3.1: Radiometric Quantities and Units

Radiant energy Q describes the total amount of light energy. Radiant flux Φ is the time rate of radiant energy. Irradiance E measures the incoming power per unit area on a surface. Radiant intensity I expresses the power emitted in a specific direction per unit solid angle. Radiance L, a central quantity in rendering, measures the power traveling in a given direction per unit area and per unit solid angle. These quantities are essential for defining physically based light transport models, including the Bidirectional Reflectance Distribution Function (BRDF), which relies directly on luminance and illuminance.
To simulate lighting as perceived by the human eye, we must relate these radiometric concepts to photometric quantitiesthose weighted by human visual sensitivity. Photometry introduces analogous quantities such as luminous flux and illuminance, which are derived from their radiometric counterparts by applying the photopic response function. This connection is crucial when using measured lighting data or photometric data in rendering systems.

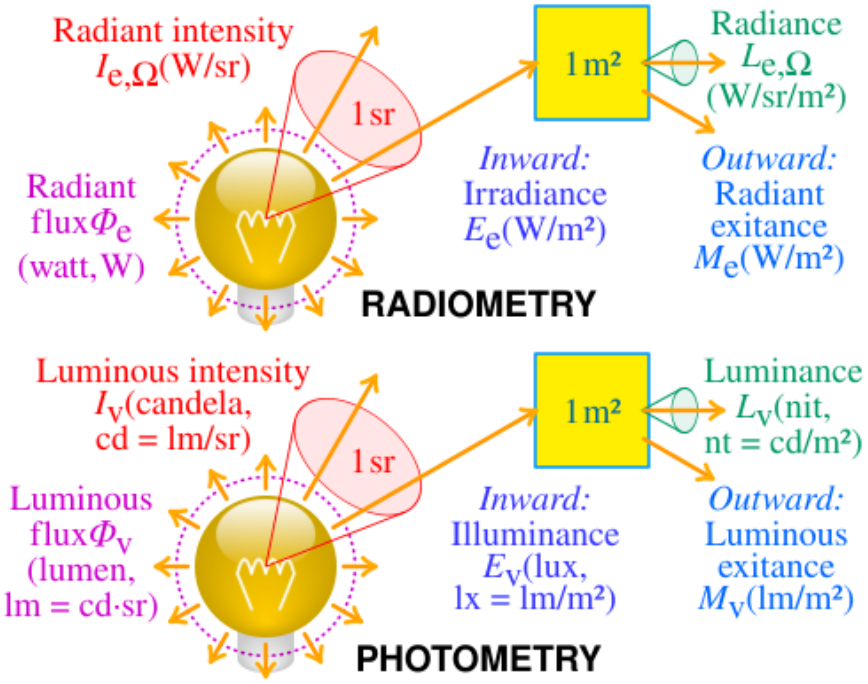


Figure 3.1: Basic units and concepts of radiometric and photometric measurements. Adapted from [11]

To model the behavior of light in rendering systems, two fundamental theoretical

frameworks are commonly referenced: physical optics and geometric optics. These models arise from different approximations of electromagnetic theory and are chosen based on the level of fidelity and computational efficiency required. In real-time rendering, the physical principles used by the simulation are primarily geometric optics [12]. It ignores the wavelength of light (considered as $\lambda \ll l$), and with this approximate treatment the laws of optics can be described in geometric terms. The geometrical optics model remains complex, and in computer graphics certain simplifying assumptions are made, which limit effects like diffraction. These assumptions are relatively simple to manage and meet the requirements of general graphics. These assumptions include:

- Object surfaces are absolutely smooth (actual roughness is modeled by microfacet meta-techniques)
- Light has only three behaviors: emission; reflection; propagation
- Light travels in a straight line at infinite speed

These assumptions allow for the efficient simulation of light in real-time environments while still producing visually plausible results. They also align with the use of physically based models that operate on radiometric and photometric principles. However there are still some shortcomings in geometrical optics based rendering, such as its inability to map structural colors accurately like the iridescence of soap bubbles in daylight. Therefore, some scholars have attempted to derive a light model in rendering by integrating physical optics. In 2020 Xia *et al.* proposed a scattering model for mapping fibers based on wave optics [13].

3.1.2 Rendering equation: foundation of light transport

In computer graphics, the rendering equation is an integral equation and serves as the theoretical foundation for all global illumination methods, including ray tracing, path tracing and radiance. Introduced to computer graphics by Kajiya in 1986 [14]. The goal of realistic rendering in computer graphics is to solve this equation. The *Rendering Equation* mathematically describes light-surface interactions:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_r(\mathbf{x}, \omega_o), \tag{3.1}$$

where L_r is the contribution of reflected light:

$$L_r(\mathbf{x}, \omega_o) = \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i \tag{3.2}$$

where:

- L_o: Outgoing radiance
- L_e: Emitted radiance (direct light sources)
- $f_r(\mathbf{x}, \omega_i, \omega_o)$: Bidirectional Reflectance Distribution Function (BRDF), which will be further described in section 3.3.

- $\mathbf{n} \cdot \omega_i$: Cosine term for geometric attenuation

The equation can be paraphrased as follows: the intensity of light reflected from point x forward the ω_o direction is equal to its own luminous intensity plus the sum of the intensity of the point's reflection of all the surrounding light in the ω_o direction. The ability of the point to reflect light incident in the ω_i direction to the ω_o direction, which is dictated by the properties of the object itself, is described by the BRDF $f_r(\mathbf{x}, \omega_i, \omega_o)$.

3.1.3 Light interaction

Light interaction is the process of light interacting with surfaces and other objects in a geometric optics model can be described as follows [15]:

1. Light is emitted from a light source.
2. The light interacts with objects in the scene, being partially reflected, partially absorbed and possibly scattered from the surface of the object in other directions after traveling a certain path.
3. Finally, the light is absorbed by a sensor (e.g., a camera or the human eye) to form an image.

3.1.4 Importance sampling

Importance sampling is a technique used to reduce variance in Monte Carlo integration. It achieves this by employing a probability density function (PDF), denoted as $p(x)$, that strategically concentrates samples in regions of the integration domain where the integrand, $f(x)$, has the most significant contributions to the integral's value [16].

The fundamental estimator in importance sampling provides a way to approximate the expected value of $f(x)$, which is corresponds to the integral of $f(x)$ over its domain, provided that $p(x)$ is the probability density function governing the sampling process. It is given by:

$$I = \int f(x)dx = \int \frac{f(x)}{p(x)} p(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(\omega_i)}{p(\omega_i)} \tag{3.3}$$

This estimator is used to calculate an approximate numerical value for the definite integral I.

where:

- N: is the total number of samples drawn.
- f: represents the target integrand. In the context of rendering, this is typically the product of the Bidirectional Reflectance Distribution Function (BRDF) and the incident radiance.

- $p(\omega_i)$: is the value of the chosen PDF evaluated at the specific sample direction ω_i. This PDF $p(x)$ (or $p(\omega)$ when integrating over directions) defines the probability density of selecting a particular sample x (or direction ω_i) from the integration domain. A higher $p(x)$ for a given x means that x is more likely to be chosen as a sample.

A PDF $p(x)$ specifies the relative likelihood of generating any given sample x. The core idea behind importance sampling is that different regions of the integration domain, such as various light directions or different locations on a surface, contribute unequally to the final integral result. The weight term, $\frac{f(\omega_i)}{p(\omega_i)}$, acts as a crucial compensating factor. It adjusts for any bias introduced by sampling according to $p(x)$ instead of uniformly, ensuring that the estimator remains unbiased. This weighting also prioritizes rays that have more significant contributions to the integral by giving them a larger effective weight if $p(\omega_i)$ is smaller than $f(\omega_i)$ and vice-versa.

For importance sampling to be effective, the PDF $p(x)$ should ideally be proportional to the absolute value of the integrand, $|f(x)|$. In rendering, this usually means that significance sampling techniques are applied to match the PDF to the directional distribution of the BRDF and/or the spatial and directional distribution of incident illumination.

This strategy is particularly crucial for simulating complex light transport phenomena where certain ray paths, such as those involving specular reflections or shadow rays directed towards bright light sources disproportionately influence the final rendered image. By sampling these important paths more frequently, the overall noise (variance) in the image can be significantly reduced for a given number of samples.

3.2 Color space

Color perception in humans is a psychophysical process that results from the interaction of incident light with the eye's photoreceptor and subsequent neural processing. To simulate this perceptual phenomenon in digital imaging systems, color must be represented using mathematical models known as color spaces. These models define numerical mappings of color stimuli, enabling consistent storage, processing, and reproduction of color across different devices and applications.

Color spaces are essential for modern rendering engines, as they facilitate the accurate translation of physical light interactions into pixel values that can be displayed on screens. Standardized color spaces such as sRGB ensure interoperability and visual consistency. For instance, to render realistic scenes, engines must convert physical quantities like radiance into screen-space RGB values using tone mapping and color space transformations.

A fundamental concept in this context is *relative luminance*, which quantifies the perceived brightness of a color by weighting its RGB components according to the spectral sensitivity of the human eye [17]. For RGB color spaces adhering to BT.709 primaries [18], relative luminance Y is computed as:

$$Y = 0.2126R' + 0.7152G' + 0.0722B' \tag{3.4}$$

where R', G', and B' are gamma-corrected RGB values. This measure is critical for tone mapping, exposure control, and brightness adaptation in rendering.

Moreover, the use of color spaces is deeply connected to photometry. In rendering pipelines, photometric quantities like luminance and radiance are computed based on physical light models and are then converted to displayable RGB values through the Bidirectional Reflectance Distribution Function (BRDF). BRDFs, central to light modeling in graphics, depend on accurate photometric calculations to simulate material appearance under varied lighting conditions.

3.3 BRDF

Achieving a realistic night driving experience in a simulator fundamentally depends on accurately simulating how light interacts with virtual object surfaces. A crucial component in this simulation is the Bidirectional Reflectance Distribution Function (BRDF). The BRDF mathematically describes the reflectance properties of a surface, quantifying the proportion of incident light from a given direction (ω_i) that is reflected towards a specific outgoing direction (ω_o). Essentially, it defines the intrinsic appearance of a material under varying lighting and viewing conditions.

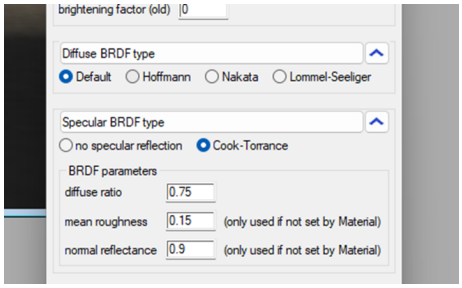
During nighttime driving, visual perception relies heavily on interpreting subtle light variations. Limited light sources such as headlights, streetlights, and ambient urban glow illuminate surfaces such as roads, vehicles, signs, vegetation, and buildings. The manner in which these surfaces reflect this scarce light, as dictated by their respective BRDFs, critically determines their appearance and contributes significantly to the overall scene realism.

Physically-based rendering (PBR) systems use BRDF models to accurately simulate these interactions. One widely adopted microfacet BRDF model is the Cook-Torrance model [19]. This model treats a surface as a collection of microscopic facets and describes the reflection by combining two components: diffuse reflection, often modeled as Lambertian, which describes light scattered uniformly, and specular reflection, which refers to mirror-like reflections concentrated around a specific direction. It allows materials to be simulated with varying degrees of diffuse and specular characteristics. The Cook-Torrance BRDF is frequently used in model materials due to its versatility and is formulated as:

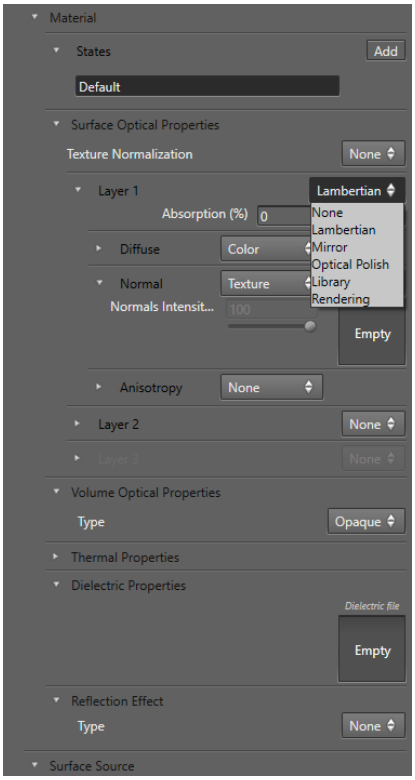
$$f_{\text{cook-torrance}} = \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \tag{3.5}$$

Here, D represents the Normal Distribution Function (NDF) describing the orientation of the microfacets, F is the Fresnel equation governing reflectance at different angles, and G is the Geometry function accounting for microfacet self-shadowing

and masking. As illustrated in Figure 3.2a, the Cook-Torrance BRDF is utilized within the LucidDrive simulator, providing controls for its parameters.



(a) The settings interface of Cook-Torrance BRDF in LucidDrive



(b) The settings interface of BRDF in AVxcelerate.

Figure 3.2: BRDF setting in AVxcelerate amd LucidDrive

Another influential PBR model is the Principled BRDF framework, developed by Brent Burley at Walt Disney Animation Studios. Its robust and user-friendly methodology for physically based rendering has led to widespread industry adoption [20]. This model is designed to be more versatile, consolidating numerous parameters into an intuitive set capable of representing a wide spectrum of materials without direct input of physical parameters. It typically incorporates distinct reflection and transmission components, lobes, that independently simulate different light transport phenomena. For instance, separate lobes may handle specular reflections from surface microfacets, diffuse scattering in subsurface layers, and light transmission through translucent materials. This modular architecture allows artists to intuitively combine physically-plausible light behaviors while maintaining energy conservation principles.

While analytical models like Cook-Torrance and the Principled BRDF provide powerful and flexible ways to define material appearance, the most accurate representations often come from measured BRDF data acquired from real-world material samples [16]. However, accessing comprehensive real-world BRDF datasets remains a challenge. Measuring Bidirectional Reflectance Distribution Functions (BRDFs) is a challenging task, as it requires densely sampling a four-dimensional domain en-

compassing both incoming and outgoing light directions. This process is known to be notoriously tedious and time-consuming, as highlighted by Dupuy and Jakob [21]. Consequently, physically-based analytical models remain essential tools in production rendering.

Figure 3.3 demonstrates the visual impact of employing different BRDF models and parameters, particularly evident in the appearance of metallic car paint under low-light conditions.

(a) Measured BRDF of metallic material

(b) Cook-Torrance BRDF (roughness 0.1 diffuse ratio 0.5 normal fresnel 0.04)

(c) Principled BRDF (metallic 0.7 specular 0.6 roughness 0.2)

(d) Measured BRDF of diffuse material

Figure 3.3: Preview with different BRDF, measured data from [21]. Render by Mitsuba 3 [22].

To ensure the night driving simulator delivers a visually accurate and perceptually experience, it is essential to carefully incorporate physically-based BRDFs, analytical models like Cook-Torrance and principled, or measured data into the simulator pipeline.

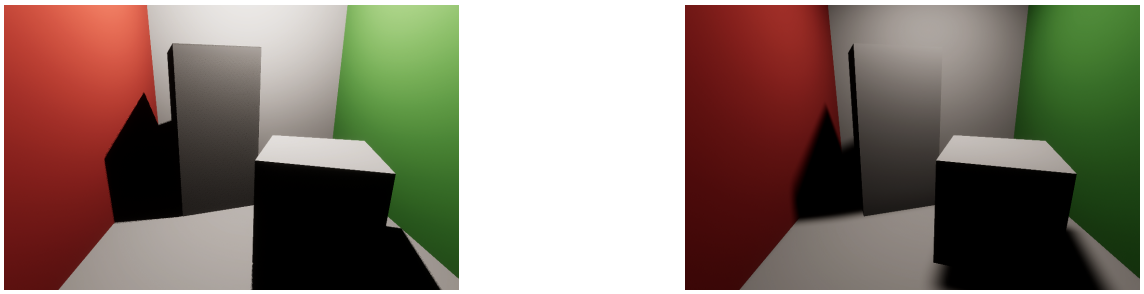
3.4 Pre-rendering and real-time rendering

The visual fidelity of a simulation is heavily influenced by the chosen rendering technique. There are two main approaches: real-time rendering, which prioritizes interactivity and speed, and pre-rendering (or offline rendering), which focuses on achieving the highest visual quality without strict time constraints. These approaches differ in their handling of complex effects such as shadows and global illumination.

3.4.1 Real-time rendering

3.4.1.1 Real-time shadow

In basic real-time rendering, shadows are often represented as 'hard shadows', shadows with sharp, well-defined edges and no gradual transition, also known as penumbra. These are typically produced using shadow mapping techniques and do not reflect the nuanced soft shadows found in real-world lighting scenarios [23]. Hard shadows generally arises from precise light sources, those with no area, only produce areas that are completely covered by shadows. Additionally, Figure 3.5 highlights a common limitation in some real-time systems where certain light sources might not



(a) (b)

Figure 3.4: The difference between (a) hard shadow and (b) soft shadow

cast shadows at all due to performance optimizations or implementation constraints. This lack of shadows from some light sources can be in stark contrast to the shadows cast by other directional light sources.

Figure 3.5: Shadows under the light source (highlighted in red box). The left side shows a shadow under a direction light (sun), while the right side illustrates a night scenario where there is no shadow under the street light.

More advanced techniques involve real-time ray tracing for shadows. While these techniques can produce physically accurate soft shadows, often achieved using area lights or by adjusting light radius parameters for point lights, ray-traced shadows are computationally demanding. This significantly impacts performance and requires powerful hardware. Moreover, even point light sources generate unnaturally sharp ray-traced shadows by default, necessitating specific parameters such as light source size to achieve realistic softness.

3.4.1.2 Real-time shadow with global illumination

Global illumination refers to the simulation of both direct and indirect light interactions in a scene. It is particularly important in fields that require dynamic lighting, especially in open-world or complex indoor scenes. Advanced ray tracing techniques are used to compute global illumination (GI). By recursively tracing rays as they bounce around a scene, these methods capture indirect lighting effects. Majercik *et al.* [24] propose a dynamic diffuse global illumination method based on ray tracing to dynamically generate irradiation field data. This method update the irradiation distribution of the scene in real time. Combined with spatio-temporal filtering or

sparse updating strategy, the irradiation is recalculated only for the changing areas of the scene, which reduces the computational overhead. This approach aims to solve the problem of indirect illumination computation for dynamic scenes in real-time rendering, particularly for dynamic illumination, in open-world or complex indoor scenes.

Direct light striking an object's surface is reflected and refracted, then interacts with other surfaces to varying degrees based on the material's color. Global illumination rendering simulates how natural light bounces off surfaces, contributing to indirect illumination, beyond just the direct illumination from light sources [25]. Section section 5.1 will present a comparison between environments rendered with and without global illumination.

3.4.2 Pre-render

Pre-rendering, also known as offline rendering, prioritizes visual fidelity over speed. This approach is used in scenarios where real-time interactivity is not a requirement during image generation, such as in film production or for creating high-quality visualizations that aim to approximate ground truth.

The process typically involves detailed object modeling, including representations for points, surfaces, and textures. This is followed by complex calculations that simulate light transport throughout the scene, considering defined materials, light sources, viewpoints, and potentially motion trajectories. For accurate simulation of global illumination effects in pre-rendering, techniques such as Monte Carlo ray tracing and its variants like path tracing are commonly employed. These methods stochastically model the complex transport of light throughout a scene based on physical principles, enabling the realistic depiction of phenomena including physically-based soft shadows from area light sources, complex diffuse and specular inter-reflections, and caustics [16].

3.5 Post-processing methods for improved rendering

The rendering process fundamentally involves sampling a 3D scene to generate a 2D image. According to the Nyquist-Shannon sampling theorem, perfect reconstruction of a sampled signal requires that the highest frequency component of the original signal be finite and sampled at a rate greater than or equal to its Nyquist frequency [26].

However, a 3D scene is a continuous function in space, encompassing its geometric coverage relations, coloring parameters, and rendering equations. This continuous function inherently lacks a finite bandwidth. Consequently, due to the scene's continuous nature and thus infinite frequency content, it is impossible to perfectly recover the original signal through discrete sampling, regardless of the sampling rate, such as image resolution.

The pixels ultimately displayed form a discrete two-dimensional array, where the determination of point coverage is a binary 'yes' or 'no' decision. This quantization leads to a loss of spatial continuity, manifesting as jagged edges visible pixelated edges that disrupt smooth lines or curves in an image. This phenomenon, often termed aliasing, is an inherent and inescapable challenge in discrete rendering systems.

Figure 3.6: A topical aliasing photo. Notice that there are artifact patterns close to Volvo letter.

3.5.1 Anti-aliasing methods

In light rendering, moiré patterns and flickering caused by illumination usually appear on surfaces with fine textures or complex geometry, and insufficient sampling causes these patterns to aliasing at the pixel level when the light is shone at a specific angle. Anti-aliasing effectively reduces these visual artifacts by increasing the effective sampling rate and blending at the pixel level. To address moiré artifacts caused by the interaction between lighting and fine texture details, Supersample Anti-Aliasing (SSAA) increases sampling resolution for both color and lighting intensity, producing smoother and more visually accurate results. Temporal flickering, such as flickering highlights caused by animation or viewpoint changes, can be minimized by SSAA, as it performs multiple spatial samples within each frame [27]. SSAA requires additional computation due to the increased sampling.

3.5.2 Denoising method

Ray tracing, and especially path tracing, often suffers from image noise due to the inherent stochastic nature of the rendering process. This noise becomes particularly noticeable when the number of samples per pixel is limited, which is often the case in performance-constrained environments. During rendering, only a finite number of light transport paths are traced per pixel to estimate the radiance reaching the camera. Because these estimates are based on random sampling of the light paths,

they exhibit high variance, which manifests visually as grainy or speckled artifacts in the final image. Unlike aliasing artifacts that occur along geometric edges and can be addressed with traditional anti-aliasing techniques, this kind of noise is distributed throughout the image in a non-uniform, often random fashion.

To address this issue, denoising algorithms are employed as a post-processing step. These algorithms aim to reconstruct a noise-free image by leveraging information from spatially and temporally neighboring pixels. Modern denoising approaches typically utilize auxiliary data such as surface normals, depth maps, albedo, and material properties to guide the filtering process, enabling more accurate differentiation between noise and actual scene details. Advanced methods, including machine learning-based denoisers, can further improve quality by learning complex mappings from noisy inputs to clean outputs. This denoising step is critical for achieving production-level image quality without incurring the computational cost of tracing an impractically high number of samples per pixel.

3.6 Structural similarity index

SSIM is a well-regarded perceptual metric used for measuring the similarity between two images, often applied in image quality assessment [28]. Unlike simple pixel-wise differences, SSIM evaluates images based on three components: luminance, contrast, and structure, aiming to quantify differences between images while considering human visual perception. This metric operates as a full-reference approach, necessitating the comparison of a test image against a reference image, which is presumed to be distortion-free or serve as a baseline.

The SSIM function is defined as follows:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.6}$$

where:

- μ_x and μ_y are the mean intensities of images x and y.
- σ_x^2 and σ_y^2 are the variances of images x and y.
- σ_{xy} is the covariance of images x and y.
- C_1 and C_2 are constants to stabilize the division.

The value of SSIM is bounded between -1 and 1. When two pictures are exactly the same, the SSIM value will be 1, which signifies a strong positive correspondence. Conversely, if one picture is the negative color of another, the SSIM will be -1, indicating a strong negative correspondence.
In this study, SSIM is used to quantify the visual similarity of light distributions rendered by different software or under different rendering configurations, providing an objective and interpretable quality metric.

The standard formulation of the SSIM operates exclusively on grayscale images, thereby neglecting chromatic information. This limitation can be significant in scenarios where color fidelity is a critical component of perceived image quality. To address this shortcoming, an extended approach is commonly adopted in which SSIM is computed independently for each of the three color channelsRed, Green, and Blue. The overall similarity score is then obtained as a weighted average of the SSIM values across these channels:

$$\mathrm{SSIM}_{\mathrm{RGB}} = w_R \cdot \mathrm{SSIM}_R + w_G \cdot \mathrm{SSIM}_G + w_B \cdot \mathrm{SSIM}_B \tag{3.7}$$

Here, SSIM_R, SSIM_G, and SSIM_B denote the SSIM values computed for the red, green, and blue channels, respectively, while w_R, w_G, and w_B are their corresponding weights. In many practical applications, equal weighting is employed (i.e., $w_R = w_G = w_B = \frac{1}{3}$). However, perceptually motivated weightings, which take into account the human visual systems varying sensitivity to different wavelengths, can also be applied to achieve more accurate assessments of perceptual quality [29].

This RGB-channel extension of SSIM enables a more comprehensive evaluation of similarity in full-color images and is widely utilized in image quality assessment tasks [30]. In this thesis, SSIM for color images is computed using equal weighting across the RGB channels unless otherwise specified.

This page was intentionally left blank.

4

Methods

4.1 Ray tracing

4.1.1 Traditional Ray Tracing

Ray tracing is a rendering technique that simulates the physical behavior of light to produce highly realistic images. It works by tracing the path of rays from the camera into the scene, recursively calculating light interactions such as reflection, refraction, and shadows. Path tracing, a Monte Carlo variant of ray tracing, further improves realism by stochastically sampling many possible light paths to approximate global illumination. The fundamental algorithm is outlined in Algorithm 1.

While path tracing is theoretically unbiased and converges to the physically correct solution given enough samples, it suffers from high variance at low sample counts, manifesting as noise in rendered images [16]. This makes it computationally expensive, especially in complex lighting conditions such as indirect illumination, caustics, or scenes with small bright light sources.

To make ray tracing feasible in real-time applications, traditional approaches often employ post-processing denoisers or importance sampling techniques. However, these methods typically rely on heuristics or precomputed data, and can introduce bias or artifacts, particularly in dynamic or disoccluded regions of the scene.

This is where ReSTIR offers a significant improvement.

4.1.2 ReSTIR

Sample reducers are techniques designed to improve the efficiency and quality of rendering by intelligently selecting, filtering, or reusing samples to reduce noise without increasing computational cost. Traditional reducers, such as spatiotemporal denoising filters, often rely on heuristics and introduce bias by discarding valuable sample information.

ReSTIR (Resampled Importance Sampling with Temporal and Spatial Reuse) builds on this concept by enabling unbiased sample reuse across pixels and frames to achieve high-quality rendering under strict real-time constraints [31]. Algorithm 2 show the outline of ReSTIR. Unlike conventional post-processing denoisers, which typically introduce bias by filtering final pixel values, ReSTIR maintains the integrity of sample

distributions by aggregating and resampling data through weighted memory layers. This allows ReSTIR to effectively 'filter' the probability density function (PDF) at the sampling stage, reducing noise without compromising physical accuracy.

The technique achieves efficiency by streaming and reusing samples directly on the GPU, avoiding the need for learning-based or precomputed sampling distributions. While grounded in previous sampling strategies, ReSTIR introduces a scalable, unified framework suitable for real-time applications. In urban nighttime environments, which feature significant geometric and lighting complexity, ReSTIRs inert sampling and memory layer merging techniques scale well without the overhead of light maps or path-guiding data structures.

When combined with a denoiser, ReSTIR can produce cleaner images with fewer rays per pixel than traditional path tracing. Its temporal and spatial resampling enables better scene coverage, though care must be taken in rapidly changing or disoccluded regions, where artifacts may occur [32]. Recent variants like Area ReSTIR [33] aim to improve antialiasing and defocus rendering through enhanced coherence handling.

While traditional path tracing remains theoretically unbiased and converges with enough samples, ReSTIR introduces a controlled bias through resampling. This trade-off often leads to faster convergence to perceptually plausible results in complex scenes. For applications that require strict physical accuracy, traditional high-sample methods may still be preferred, but ReSTIR continues to evolve toward narrowing this gap.

Algorithm 1 Monte Carlo Ray Tracing

Input: Camera, Scene, Max Depth
Output: Image
for each pixel in the image **do**
  $ray \leftarrow$ shoot ray from camera to pixel
  $color \leftarrow$ trace(ray, Scene, Max Depth)
  set pixel color to *color*
end for
Function: trace(ray, Scene, Depth)
if Depth > Max Depth **then**
  return black
end if
$intersection \leftarrow$ find intersection of ray with scene
if intersection exists **then**
  $color \leftarrow$ compute shading at intersection
  $reflection \leftarrow$ trace reflection ray
  $refraction \leftarrow$ trace refraction ray
  $color \leftarrow color + reflection + refraction$
  return color
else
  return background color
end if

Algorithm 2 ReSTIR Algorithm

Input: Camera, Scene, Max Depth, Num Frames
Output: Image
initialize spatiotemporal samples buffer
for each frame in Num Frames **do**
  for each pixel in the image **do**
    $ray \leftarrow$ shoot ray from camera to pixel
    $sample \leftarrow$ trace(ray, Scene, Max Depth)
    store sample in spatiotemporal buffer
  end for
end for
for each pixel in the image **do**
  $sample \leftarrow$ resample samples from buffer
  set pixel color to average resampled sample
end for
Function: trace(ray, Scene, Depth)
if Depth > Max Depth **then**
  return black
end if
$intersection \leftarrow$ find intersection of ray with scene
if intersection exists **then**
  $color \leftarrow$ compute shading at intersection
  $reflection \leftarrow$ trace reflection ray
  $refraction \leftarrow$ trace refraction ray
  $color \leftarrow color + reflection + refraction$
  return color
else
  return background color
end if

4.2 Research design

This section details the experimental framework and analytical methods employed to evaluate light simulation accuracy and performance across rendering platforms. The methodology follows a three-phase structure: data acquisition, rendering pipeline configuration and qualitative validation.

The study employs a comparative approach to evaluate the trade-off between real-time and pre-rendering in low global illumination, cross-software consistency of light distribution and the impact of different light-tracing algorithms, antialiasing on rendering quality, hardware consumption

A hybrid workflow combines physically-based rendering (PBR) principles with perceptual metrics (SSIM) and hardware performance profiling. The Cornell Box serves as the primary test scene due to its standardized geometry, materials and controlled lighting conditions [34]. The Volvo car model of XC60 will be used for feasibility verification of vehicle scenarios.

All rendering performed with i7-11850H, RTX A3000 GPU.

4.2.1 Data acquisition and preprocessing

All data in this work relating to photometry are in IES format. IES files for headlamps and streetlights obtained from automotive Original Equipment Manufacturer datasets. To accurately model vehicle lighting, simulators often utilize IES (Illuminating Engineering Society)[1] photometric data files. An IES file is a standardized plain text format describing the intensity distribution of a light source in 3D space, providing luminous intensity values at various vertical and horizontal angles relative to the source [35].

4.2.2 Image alignment methodology for light distribution maps

Accurate alignment of light distribution maps is crucial for comparative analysis in rendering quality assessments, particularly when evaluating factors like global illumination, denoising effectiveness, or the impact of advanced sampling techniques. Geometric discrepancies, often introduced by slight camera movements, scene adjustments, or rendering pipeline variations, necessitate a robust image registration process. This section details the methodology employed for aligning light distribution maps, ensuring spatial correspondence for subsequent quantitative and qualitative evaluations.

The alignment process is structured into three primary stages: image preprocessing, feature detection and matching, and robust geometric transformation estimation.

[1] Another possible file format that may be used is EULUMDAT.

4.2.2.1 Image preprocessing and feature extraction

The first step is to convert the input images to a suitable format for feature extraction and enhancement. Since light distribution maps can have varying luminance levels and dynamic ranges, the images are initially converted to grayscale based on their relative brightness. This grayscale representation forms the foundation for further processing. Next, the gradient magnitude of the grayscale image is calculated. This operation emphasizes areas with significant intensity changes, which are essential for identifying distinctive features. Using the gradient magnitude map, the Scale-Invariant Feature Transform (SIFT) algorithm is then applied. SIFT is selected for its robustness to changes in image scale, rotation, and illumination [36].

To explain the flow of image preprocessing, assume that a reference image, denoted as image 1, and an image to be aligned, image 2, SIFT is used to:

1. **Detect Keypoints:** Identify distinctive points in the image that are invariant to scaling and rotation, such as typically corners, blobs, or points of high contrast. In this thesis, the grayscale gradient based on the light distribution map will be used as the input for keypoint detection.
2. **Extract Descriptors:** Compute a unique descriptor for each detected keypoint, representing the local image gradient information around the keypoint. These descriptors facilitate robust matching between images.

This process yields sets of keypoints and their corresponding descriptors for both the reference and target images.

With feature descriptors extracted, the next stage involves establishing correspondences between image 1 and image 2. This is achieved by comparing the descriptors from image 1 with those from image 2 to find the best matches. A critical aspect of this matching process is the implementation of filtering criteria to enhance match quality:

The result of this matching step is a set of putative correspondences linking keypoints from image 1 to image 2. From these matches, corresponding source keypoints and destination keypoints are assembled.

4.2.2.2 Robust geometric transformation and image warping

Despite the rigorous matching criteria, the set of putative matches often contains outliers due to noise, repetitive patterns, or inherent ambiguities. To robustly estimate a geometric transformation that aligns image 2 with image 1, the Random Sample Consensus (RANSAC) algorithm is employed. RANSAC, an iterative method, estimates parameters of a mathematical model from a set of observed data that may contain outliers. This is crucial for ensuring accurate alignment [37].

A projective transform, also known as a homography, serves as the geometric model in this context. This general 2D planar homography corrects misalignments between images, even those with slight viewpoint changes, by accounting for translation, rotation, scaling, shearing, and perspective distortion.

RANSAC is employed to find the most robust transformation. It iteratively selects a minimum of four sample points to estimate a model and then assesses how many other data points align with this model within a specified residual threshold. The model that maximizes the number of inliers is chosen as the robust model. Additionally, RANSAC provides a boolean array, indicating which matches contributed to the final robust model.

Finally, the estimated robust transformation is applied to image 2 to align it with image 1.

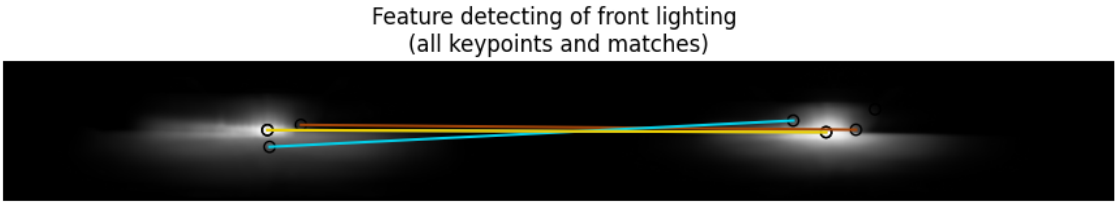


Figure 4.1: Schematic diagram of light distribution map feature detection

4.2.3 Rendering pipeline configuration

Unity was used as a third party renderer in addition to AVxcelerate and Luciddrive. Default Unity render pipeline is HDRP, with the ReSTIR algorithm added. This addition aims to confirm and measure the effect of ReSTIR algorithm on real time rendering of headlights at night. For the difference between the two pipelines, refer to Figure 4.2 and Figure 4.3. The ReSTIR component is integrated into the Lighting Pass stage in Figure 4.2.

The G-buffer (geometry buffer) is a collection of intermediate textures generated during the geometry pass of a deferred rendering pipeline. These textures store various lighting-relevant attributes for each visible pixel, such as surface normals, albedo, depth, material roughness, and metallicity. This decoupling of geometry and lighting allows for more efficient and flexible lighting computations during the subsequent lighting pass [38]. In our implementation using Unity's HDRP, the G-buffer plays a critical role in enabling real-time global illumination techniques such as ReSTIR. A significant enhancement involves compressing and storing IES light profile textures within the G-buffer itself. This storage mechanism provides direct access to IES data during shading, thereby enabling accurate rendering of the geometric light shapes defined by these profiles.

In contrast, AVxcelerate and LucidDrive are commercial closed-source tools, making it difficult to precisely identify the rendering pipeline they employ. While they are capable of producing high-quality visual outputs, especially in automotive simulation contexts, the lack of transparency in their internal architecture prevents a direct analysis or modification of their lighting or shading stages. This limitation underscores the value of using Unity as an open and modifiable reference renderer for evaluating the effect of integrating ReSTIR into the lighting pipeline.

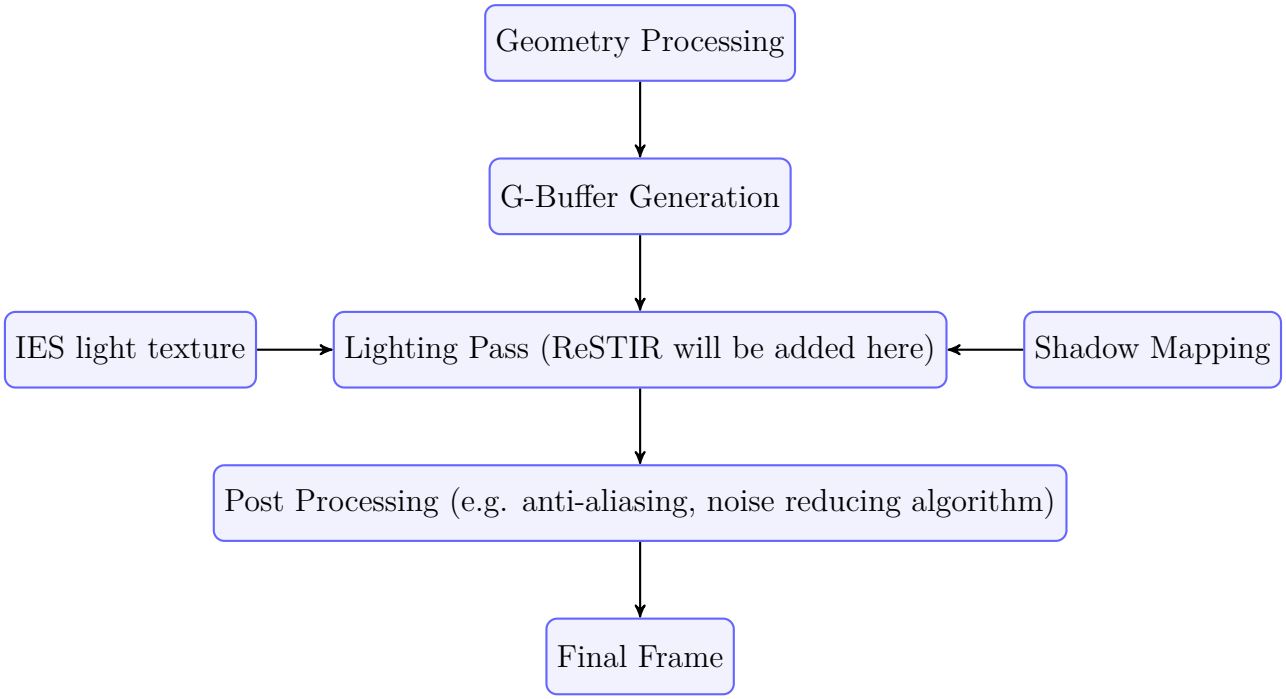


Figure 4.2: Unity HDRP Pipeline

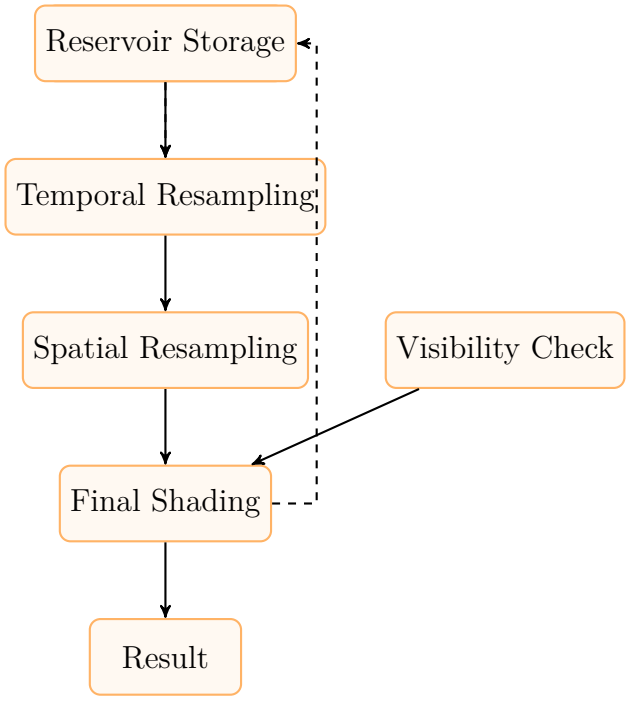


Figure 4.3: ReSTIR Pipeline

4.2.4 Perceived visual quality

Perceived visual quality refers to a human observer's subjective judgment of how visually pleasing or realistic a rendered image appears. In computer graphics, evaluating image quality can be performed using three primary categories of metrics: full-reference (FR), reduced-reference (RR), and no-reference (NR) methods.

Full-reference metrics, such as SSIM, require access to a high-quality reference image and compare it pixel-by-pixel or structurally against a distorted or approximated rendering. These metrics are widely used in evaluating the fidelity of image synthesis and noise reduction algorithms. Reduced-reference methods use only partial information from the reference image, while no-reference approaches aim to predict quality without any reference at all, often leveraging statistical priors or learned perceptual features.

In the context of this study, full-reference metrics such as SSIM are particularly relevant, as they enable quantitative comparisons between ground-truth physically-based rendered outputs and approximated outputs produced under real-time or resource-constrained conditions.

Computer graphics rendering methods often rely on complex physical simulations of light propagation, which are computationally expensive due to the need to model phenomena such as indirect illumination, global light transport, and specular-diffuse interactions. While these simulations strive for physical accuracy, many practical application including real-time rendering in games or visualization prioritize perceptually plausible results over strict physical correctness [39].

Therefore, perceptually oriented metrics like SSIM provide more meaningful insight into the visual acceptability of renderings, as they align more closely with human perception than raw numerical differences. In the results, SSIM is used to benchmark the effectiveness of the proposed rendering pipeline in preserving structural and visual fidelity. Higher SSIM scores indicate better perceptual similarity, making them a useful supplement to traditional error-based metrics in assessing rendering quality.

5

Results

This chapter presents the comprehensive results of our comparative evaluation of different simulation platforms and rendering methodologies for night-time driving environments. The analysis begins with an examination of the real-time rendering performance across three distinct software solutions, focusing on their capabilities in handling complex illumination effects and their impact on visual fidelity. Subsequently the influence of various ray tracing techniques and sampling strategies on image quality and computational efficiency is investigated. The chapter also explores critical aspects of light illumination distribution, assessing cross-software consistency and the effects of different sampling resolutions for IES files. Finally, methods for enhancing image quality, such as anti-aliasing and denoising are discussed, along with an analysis of the associated performance overhead and latency in real-time rendering.

5.1 Rendering performance for each software

5.1.1 Real-time render performance with three software

Performance evaluations were conducted using the Cornell Box scene across three rendering platforms, with real-time capabilities and illumination effects being key assessment criteria. As shown in Figure 5.1, significant differences emerge in shadow handling and indirect lighting implementation between the solutions.

LucidDrive exhibits fundamental limitations in real-time shadow computation (Figure 5.1e), while AVxcelerate employs baked shadow maps that remain static regardless of dynamic lighting changes in vehicular systems (Figure 5.1d). The comparison between Unity's standard render pipeline (Figure 5.1b) and its ray-traced global illumination implementation (Figure 5.1c) reveals enhanced inter reflections, particularly in the occluded region between the red wall and elongated box. This demonstrates improved handling of diffuse lighting and secondary light bounces through path tracing.

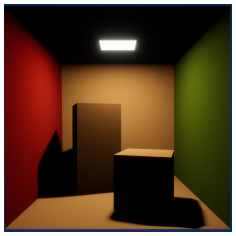
These visual differences are quantitatively supported by the SSIM values in Table 5.4. Compared to the reference image, Unity with global illumination achieves the highest structural similarity (0.7237), aligning with the observed improvement in lighting realism. LucidDrive, despite its low GPU memory footprint and consistent 60FPS performance, scores a moderate SSIM of 0.6688, reflecting acceptable color and

structure reproduction. AVxcelerate, with static lighting, scores the lowest SSIM in RGB channels (0.3803), highlighting its limited capacity to model dynamic illumination accurately. Unity without global illumination yields lower scores (0.4998 overall SSIM), consistent with its lack of indirect lighting.

It's important to note that the standard SSIM calculation typically converts images to grayscale for structural similarity assessment. To account for color reproduction, a weighted average of the SSIM values for each RGB channel was also calculated. A comparison between the weighted average SSIM and the original grayscale SSIM results revealed a consistent difference of 0.05 to 0.1 across all scores. However, this discrepancy did not alter the relative ranking of the results and therefore did not affect the overall outcome. This indicates that, for the purposes of this analysis, the grayscale SSIM provides an adequate representation of overall perceptual quality, including aspects related to color fidelity.

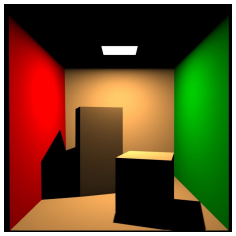
Hardware utilization metrics further emphasize the trade-off between visual fidelity and computational cost. While LucidDrive operates efficiently with only 222MB VRAM, Unity's ray-traced solution requires approximately 1GB but runs at 124FPS due to effective GPU acceleration. All platforms leverage GPU resources, though their efficiency and output quality vary considerably.

(a) Reference [40]

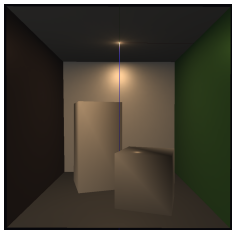


(b) Cornell box in Unity

(c) Cornell box in Unity with Global illumination



(d) Cornell box in AVxcelerate



(e) Cornell box in LucidDrive

Figure 5.1: Cornell Box in each software

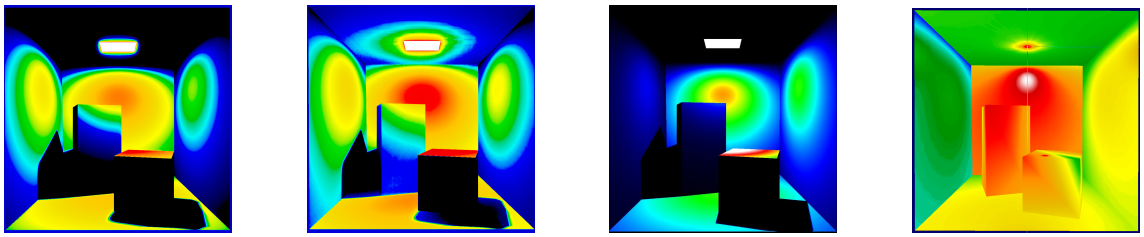
Figure 5.2 allows for further analysis of direct illumination characteristics. The ray-traced global illumination in Unity (Figure 5.2b) demonstrates physically accurate light diffusion, evidenced by soft shadow penumbras resulting from indirect lighting contributions. This contrasts with LucidDrive's incomplete illumination model (Figure 5.2d), which fails to properly account for surface irradiance.

Table 5.1: SSIM score of cornellbox compare with reference

	SSIM	SSIM in RGB channal
reference	1	0.9477
Unity without GI	0.4998	0.4294
Unity with GI	0.7237	0.6054
AVxcelerate	0.5274	0.3803
Luciddrive	0.6688	0.6320

Table 5.2: Frame rate and GPU memory consumption of each software of Cornell box

	frame rate	GPU consumption
Unity without GI	275FPS	937.9MB
Unity with GI	124FPS	1GB
AVxcelerate	62FPS	781MB
Luciddrive	60FPS	222MB



(a) Cornell box in Unity

(b) Cornell box in Unity with Global illumination

(c) Cornell box in AVxcelerate

(d) Cornell box in LucidDrive

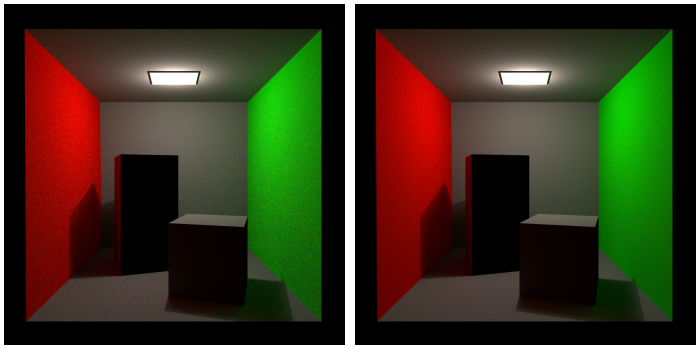
Figure 5.2: Cornell Box in each software with illumination map.

5.1.2 Offline rendering performance

Ansys and Synopsys also provide offline renders to enhance rendering quality. However, the inability to render in real-time precludes their application in nighttime driving simulations. The primary advantage is the potential for extremely high realism, irrespective of the time taken per frame. The main disadvantage is the lengthy computation time and the inability of users to interact with the scene during rendering. Listed below are some test results of Ansys offline renderer Speos in a Cornell box scene. As shown in Figure 5.3 and Table 5.3, increasing the computational effort, such as the number of passes or emitted photons in path tracing, reduces noise and improves quality but significantly increases rendering time.

Table 5.3: Rendering data of offline rendering ray tracing in Ansys Speos

	Number of emitted photons	Average number of photons per pixel	Simulation time
100 pass	1.49×10^9	1420.47	96 sec
1000 pass	1.38×10^{10}	13122	883 sec



(a) Offline render in Ansys Speos of 100 pass

(b) Offline render in Ansys Speos of 1000 pass

Figure 5.3: Offline render

5.1.3 Real-time rendering performance with different methods

To test the algorithm's impact on the real-time rendering performance, Figure 5.4 presents tests conducted using Monte Carlo ray tracing in 1 sample per pixel (SPP), 99 SPP and 99 SPP with ReSTIR. Although rendering with insufficient sample points can result in noisy image, ReSTIR can still produce high-quality outputs.

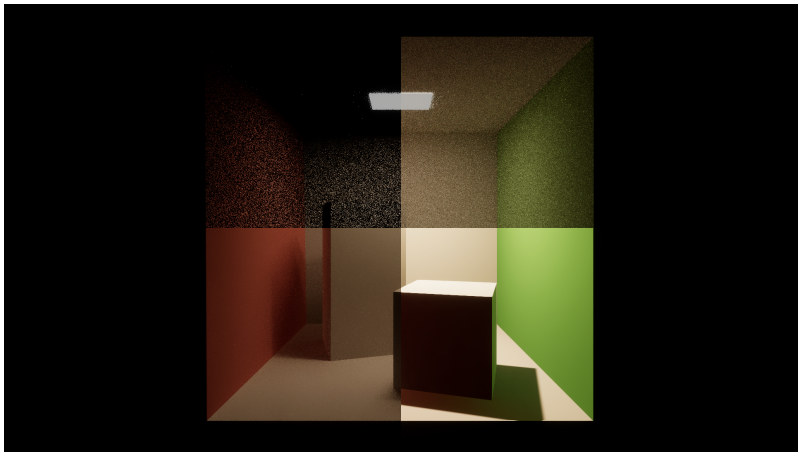


Figure 5.4: Different method testing in Unity. Left up: Ray tracing in 1 spp. Right up: Ray tracing in 99 spp. Left down: Ray tracing in 99 spp+ReSTIR. Right down: Reference

The quantitative results in Figure 5.5 illustrate the effect of sample rate and the ReSTIR technique on the visual clarity of the rendered images, as measured by the SSIM score. A very low sampling rate (1 SPP) results in poor image quality, with the SSIM score remaining low and showing minimal improvement over time. Increasing the sampling rate to 99 SPP improves quality and demonstrates gradual convergence over successive frames. However, combining the ReSTIR technique with a 99 SPP setup significantly improves initial image quality and accelerates the convergence process, achieving substantially underscores SSIM scores in fewer frames. This highlights ReSTIR as a highly effective optimization technique that markedly enhances the visual quality and convergence speed of ray tracing rendering while maintaining a reasonable sampling budget.

In global illumination research, the Cornell Box is a fundamental test scenario that comprises 77 vertices and 36 triangular facets. In contrast, automotive models demand more sophisticated geometric representations to accurately capture complex curvature features. To evaluate light transport algorithms under industrial-grade complexity, a Volvo vehicle top-hat model is employed, containing 374 877 vertices and 605 170 triangular elements. Figure 5.6 compares rendering results using different sampling strategies. The left panel shows that despite the use of 4056 SPP - significantly higher than the regular sampling requirements of the Cornell Box scene - there is still a significant amount of noise contamination. This degradation stems from the geometric complexity of the model, which results in intricate light transmission paths. The surface complexity increases the variance of the Monte Carlo esti-

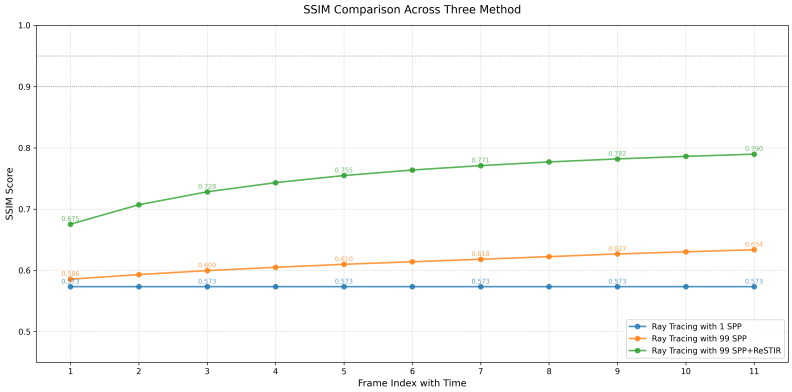


Figure 5.5: . Frames are organized in time

mator. The figure on the right demonstrates the application of ReSTIR,highlighting its effectiveness in suppressing noise on planar surfaces. However, residual artifacts remain in the concave region, as discussed in subsection 4.1.2.

Figure 5.6: Testing with Volvo XC60 model in Unity. With 4056 spp in Unity.

In terms of illuminations, a similar conclusion can be drawn, depicted in Figure 5.7. The luminance isometric map representation is most continuous in AVxcelerate. The ray tracing technique enhances quality on the concave surface, as evidenced by the variations in light and dark areas. Ray tracing calculates both direct and indirect light and thereby improving rendering realism. This approach can be useful to validate the headlight behavior accurately.

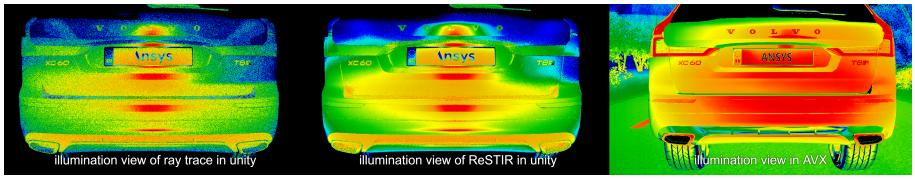


Figure 5.7: Comparison of illumination heat maps showing results from Unity (left and center) and AVxcelerate (right).

5.2 Light illumination distribution map

5.2.1 Cross-software consistency analysis using SSIM

An important observation is that importing the exact same IES file into different rendering software packages can yield visually distinct light illumination patterns on a target surface. To objectively quantify these differences, image comparison metrics, specifically the structural similarity index measure (SSIM). The testing IES file which serves both as input and reference is a measurement of a generic high beam-pattern.

The reference image is derived from the original IES file and undergoes linear conversion from light intensity to an RGB picture during testing, while the software-generated image targets a lighting-aiming wall from the same light source. This type of verification is solely based on visual inspection and assesses the illumination details in the aiming wall. The brightness of the photo is linearly scaled from illumination data to 0-255 grayscale image. Due to differences in image size, the image must be resized to a fixed size. Feature matching was performed on the gradient map of the light distribution using SIFT, and robust feature points were filtered with RASCN and projectively transformed to align them with the reference image. The most significant part is shown in Figure 5.8.

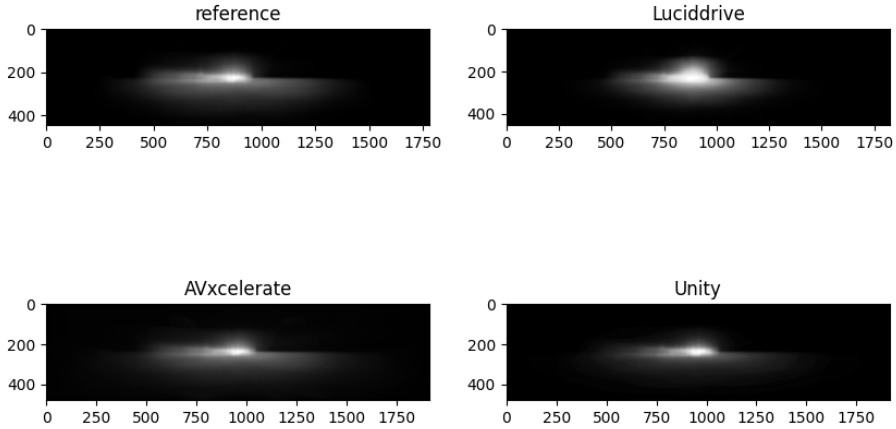


Figure 5.8: Front light distribution map in different software.

Finally, the SSIM result are visualized in Figure 5.9. The heat map, indicates that areas closer to white represent higher similarity between two images while areas closer to black indicate greater difference. Significant differences are observed in the upper edge, the lower left and right lower area.

In AVxcelerate, there are still large areas of difference in light distribution. This is due to variations backdrops and camera sensors between different software. AVxcelerate uniquely preserves a very low level of invisible light in the dark areas, rather than setting the RGB value to 0, which results in a low SSIM score for AVxcelerate. Figure 5.10 shows that although both illuminist the aiming wall, only AVxcelerate has a sensor range wide enough to collect the light from the aiming wall. When

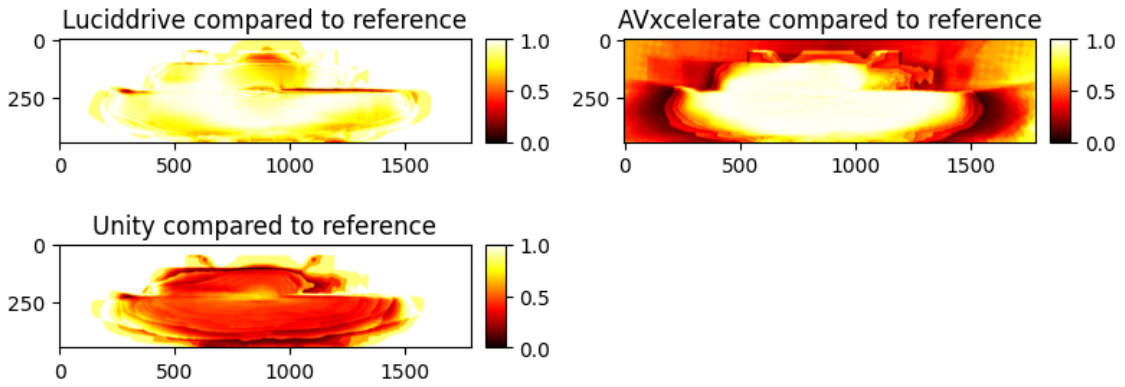
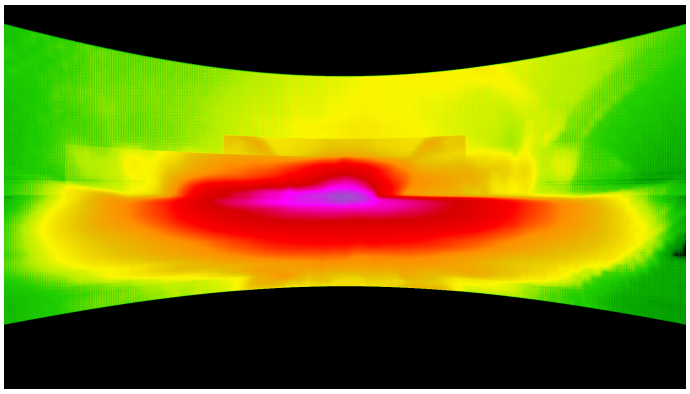


Figure 5.9: Visualize the SSIM of light distribution map in each software.

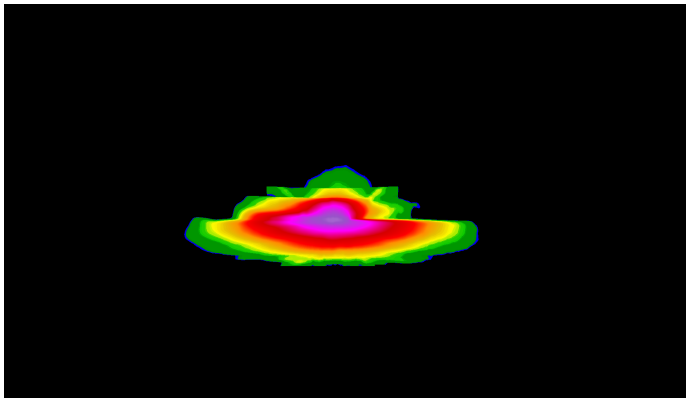
Table 5.4: SSIM score of front light distribution map compare with reference

	SSIM
Luciddrive	0.9075
AVxcelerate	0.6152
unity	0.7406

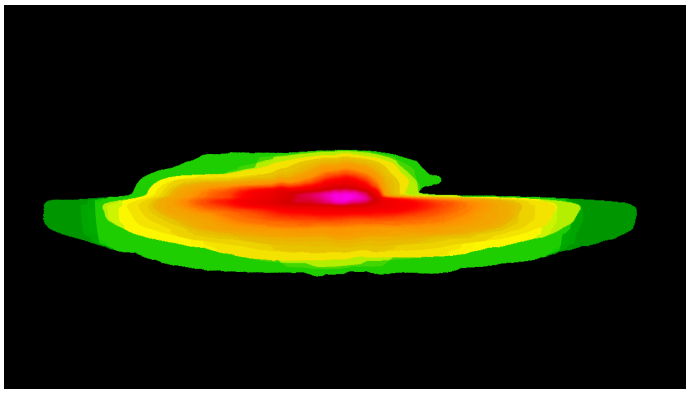
converting the IES file into light, AVxcelerate and LcidDrive perform more direct conversions from low illumination data. Shape edges in the light distribution are noticeable in Figure 5.10a and Figure 5.10b. In Figure 5.10c the distribution curve for medium brightness is smoother. Unity's direct filtering eliminates the relevant low brightness parts.



(a) Light illumination in AVxcelerate



(b) Light illumination in Luciddrive



(c) Light illumination in unity

Figure 5.10: Illumination view of light distribution in the aiming wall

5.2.2 Sampling

Beyond variations across different software, rendering results can also vary significantly within a single software environment, depending on how the IES file is sampled. In AVxcelerate, for instance, the headlamp source sampling parameter controls the angular resolution used to interpret the IES photometric data, thereby influencing the fidelity of the light distribution pattern.

To evaluate the impact of different sampling resolutions, the same IES file was rendered using two angular steps: 0.1° and 0.05°. The grayscale outputs of both renderings are shown in Figure 5.11, where the same region of the light pattern is zoomed in for direct visual comparison. Increasing the angular resolution or decreasing the step size results in sharper, more defined rendered light edges. Conversely, coarser sampling, which involves a larger step size, leads to interpolation artifacts and a less accurate appearance of the light edge.

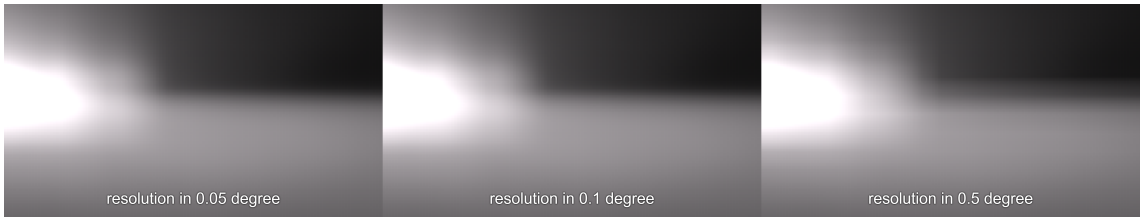
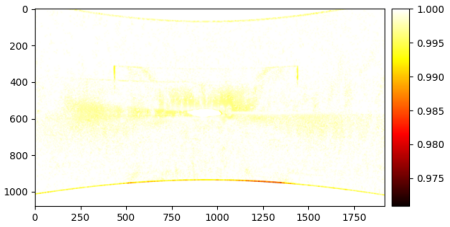


Figure 5.11: Renderings of the same light pattern in AVxcelerate with different sampling resolutions. A smaller angular step (0.05°) yields a sharper and more precise edge compared to the coarser 0.1° sampling.

To quantify the differences, an SSIM heatmap between the two renderings is presented in Figure 5.12a. The darker regions in the map indicate larger discrepancies, predominantly located along high-gradient transitions such as the beam cut-off edge. This suggests that higher angular sampling resolution primarily improves rendering fidelity in regions where the light intensity changes rapidly. On the aiming wall, some square shapes can be observed out of the main area of the light. Obversely, sampling rate could effect the compression step to handle the IES file in low illumination areas. This type of compression effect is not visible to the human eye but can cause distortion in measurable light distribution leading to inaccurate results.

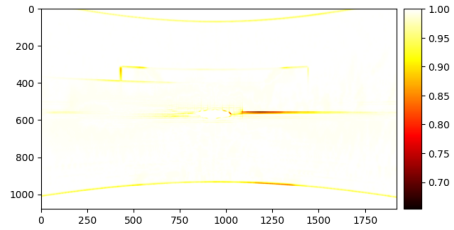
However, increasing sampling precision also comes at a computational cost. As shown in Table 5.5, higher angular resolution leads to increased GPU memory usage. For example, reducing the sampling step from 0.1° to 0.05° results in approximately 100 MB more GPU memory consumption. Conversely, using a coarse sampling step of 0.5° reduces memory usage to under 860 MB.

The experimental data demonstrates the difference in hardware consumption and rendering output required for various sampling strategies. For regulatory validation or optical benchmarking, higher angular resolution is recommended to ensure accurate reproduction of the beam shape. For nighttime simulation or driver training applications, a moderate sampling rate may be sufficient if acceptable performance can be maintained while avoiding visible artifacts.



s

(a) SSIM map comparing the 0.1° and 0.05° sampling results in AVxcelerate.



(b) SSIM map comparing the 0.1° and 0.5° sampling results in AVxcelerate.

Figure 5.12: The most prominent differences occur along sharp intensity gradients, especially at the upper light edge.

Table 5.5: GPU memory consumption with varying angular sampling resolutions in AVxcelerate

Sampling step (°)	GPU Memory Usage	GPU Utilization (%)
0.05	1.3 GB	27
0.1	1.2 GB	25
0.5	859.7 MB	25

5.2.3 Light distribution under ray tracing algorithms

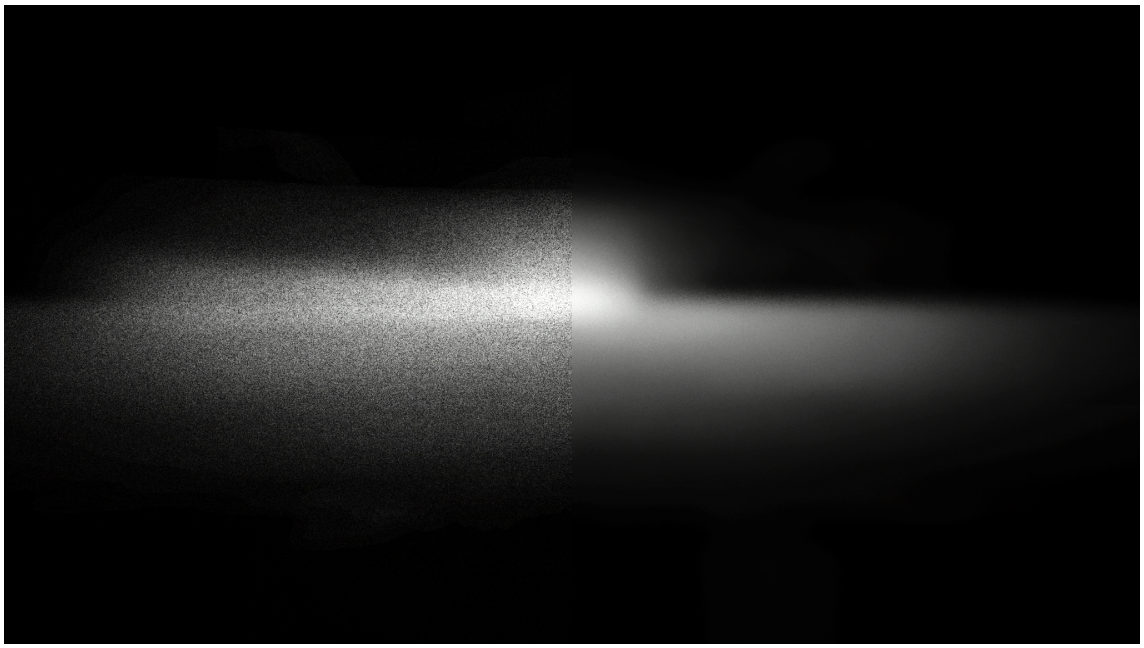


Figure 5.13: Ray tracing of front headlamp in Unity with 24 spp. Left side is traditional ray tracing and right side is ReSTIR.

In the previous section, the distribution of headlights did not use real-time ray tracing. This section will examine how the utilize of a real-time ray tracing pipeline will affect the real-time rendering of the headlights.

The ray tracing section (left of Figure 5.14) demonstrates the typical results of under-sampling during Monte Carlo rendering, resulting in excessive variance. The ReSTIR section (right of Figure 5.14) demonstrates its ability to resample information to produce sharper results. This results in more visually appealing and interpretable images, with smoother light transitions and clearer depiction of lighting patterns.

Comparing with the reference, Figure 5.15 show how this two different ray tracing applied in front light distribution.

Figure 5.14a shows a typical high noise image produced by standard ray tracing

(a) Monte-Carlo ray-tracing (b) ReSTIR

Figure 5.14: Illumination view of light distribution on an aiming wall

when there are not enough samples. In complex indirect light scenes, standard ray tracing often results in high variance with a limited number of samples. This variance manifests as noise in the image. Although the average expectation is accurate, the luminance calculated for an individual pixel or region may significantly differ from the true value until the number of samples is sufficiently large enough to smooth out these fluctuations. Consequently, at low sample sizes, the instantaneous illuminance accuracy may be poor, despite the expectation of accuracy, due to the dominance of noise.

Figure 5.14b demonstrates the result of applying ReSTIR technique, which significantly reduces the noise through efficient sample reuse and importance sampling strategies. This result in smoother and clearer images, leading to better rendering quality or faster convergence at lower computational costs. ReSTIR effectively concentrates the sampling budget on the light paths and regions that contribute the most to the image. To achieve this efficiency, some variants and implementations of ReSTIR may introduce a certain bias. This implies that even with unlimited computational resources are available to run within the framework of ReSTIR, the results may not converge perfectly to the physically true value, but rather to a slightly biased value and can arise from two main sources. Spatial resampling may lead to blurring or distortion of illumination details, affecting the accuracy of local illuminance. This occurs because the assumption of similar illumination in neighboring regions doesn't always hold perfectly, such as at shadow boundaries or with complex geometric details. Additionally, temporal resampling can result in lagging or ghosting, a form of temporal illuminance inaccuracy, especially in dynamic scenes where lighting or objects change rapidly and information from the previous frame is used.

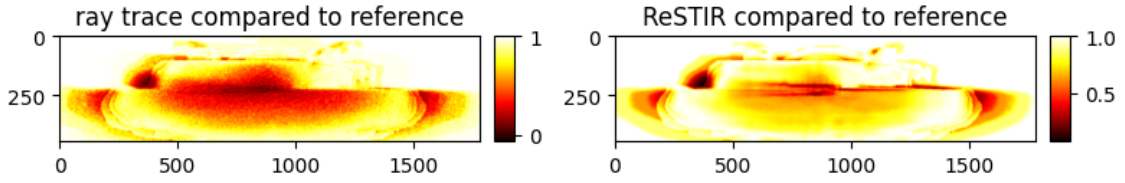


Figure 5.15: Two ray tracing algorithms producing the front light distribution map. Compare to reference, the SSIM of ray trace is 0.7625 and ReSTIR is 0.8632

For applications requiring absolute, physical measurement-level luminosity accuracy, such as scientific simulations or precise real-world comparisons, standard ray tracing is theoretically more reliable. This is because, with sufficient samples, ray tracing is inherently unbiased. The potential for bias, even if minor, introduced by ReSTIR could pose an issue when such stringent requirements are in place.

5.3 Post-processing method of real-time rendering

5.3.1 Anti aliasing

Currently among the three compared software, anti-aliasing during real-time simulation is exclusively available in AVxcelerate. Table 5.6 demonstrates the GPU resource consumption under different anti-aliasing levels while maintaining identical input texture quality and software frame rates. Figure 5.16 illustrates the nighttime rear-view visualization with vehicle lighting effects. Figure 5.17 presents detailed model edge rendering comparisons, while Figure 5.18 examines light reflection patterns on vehicle surfaces.

The comparative analysis reveals progressive improvements in edge smoothing with higher anti-aliasing levels, as particularly evident in Figure 5.17. However, as shown in Figure 5.18, surface light reflections continue to exhibit residual artifacts even at maximum anti-aliasing settings. Furthermore, it is important to note that while anti-aliasing techniques like FXAA (fast approximate anti-aliasing) can effectively smooth geometric edges [41], they generally have minimal impact on mitigating the stochastic noise points inherent in under-sampled ray tracing, as illustrated in Figure 5.19.

Table 5.6: GPU memory consumption of anti aliasing in AVxcelerate

	GPU average usage	GPU average consumption	memory consumption
None	23	2.1GB	4.3 GB
2×	25	2.3GB	4.4 GB
8×	28	2.2GB	4.0 GB
16×	33	2.4GB	4.2 GB

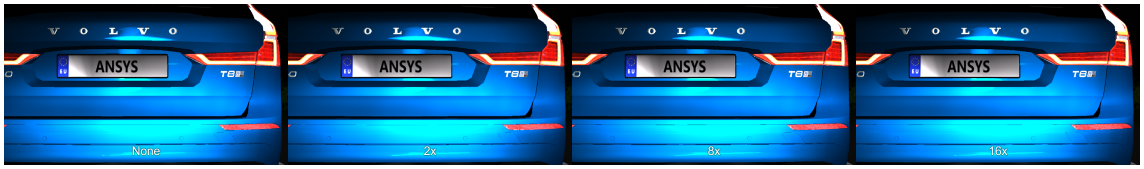


Figure 5.16: Anti aliasing in detail. From left to right is None, 2×, 8× and 16×

Figure 5.17: Anti aliasing in detail. From left to right is None, 2×, 8× and 16×. Zoom in to the line of the model shape.

Figure 5.18: Anti aliasing in detail. From left to right is None, 2×, 8× and 16×. Zoom in to light spot of the car.

Although these algorithms are highly effective at their intended purpose of reducing aliasing, their efficacy in addressing image noise is notably limited. Noise refers to unwanted random variations in pixel intensity or color that are not part of the intended image signal. These algorithms are designed to address coherent, predictable patterns caused by under-sampling geometric primitives. Noise, on the other hand, is characterized by its random and incoherent nature. When an anti-aliasing algorithm averages noisy pixels, it may slightly diffuse the noise, but it does not effectively remove the underlying random fluctuations. In fact, some anti-aliasing techniques, particularly those relying heavily on temporal accumulation, such as temporal anti-aliasing, can sometimes introduce new artifacts like ghosting or smearing if not carefully implemented [42]. These artifacts can be perceived as a different form of visual distraction rather than a reduction in existing noise. Figure 5.19 shows how anti-aliasing imparts in ray tracing denoising. The effect is not significant.

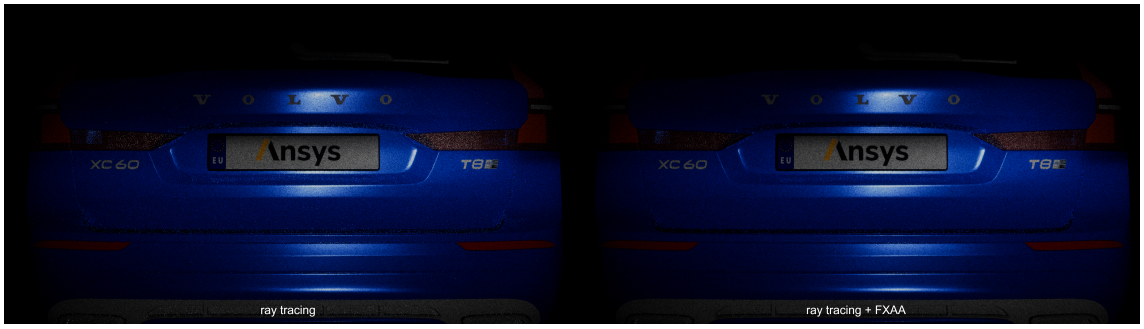


Figure 5.19: Anti aliasing in Unity. Comparing traditional ray tracing and ReSTIR.

5.3.2 Denoising

Denoiser are designed to remove noise from the rendering result by using noise reduction techniques to make the rendering result truly acceptable to the human eye. Of course, the introduction of noise reduction algorithms also needs to take into account the additional performance overhead of noise reduction. As illustrated in Figure 5.20, the visual output of standard Monte Carlo ray tracing is compared against two enhancement techniques: a post-processing denoiser and ReSTIR. The leftmost panel, representing baseline ray tracing, exhibits significant stochastic noise, or speckle, which is characteristic of an undersampled rendering process. In the middle panel, the application of a denoiser substantially mitigates these artifacts. By filtering the noisy input, the denoiser produces a visually cleaner and more coherent image, though some loss of fine detail or blurring can be a potential side effect.

The rightmost panel demonstrates the efficacy of ReSTIR. This technique improves the underlying sampling process by reusing information from adjacent pixels and previous frames, leading to a rendered image with inherently lower variance. As shown, ReSTIR can produce a result that is significantly cleaner than the raw output and often superior to what is achieved by the denoiser alone. The fundamental distinction is that a denoiser is a remedial post-process that cleans existing noise, whereas ReSTIR is a sampling acceleration technique that prevents much of the noise from being generated in the first place. This makes ReSTIR a powerful method that can also be used synergistically with a denoiser to achieve exceptionally high-quality results.

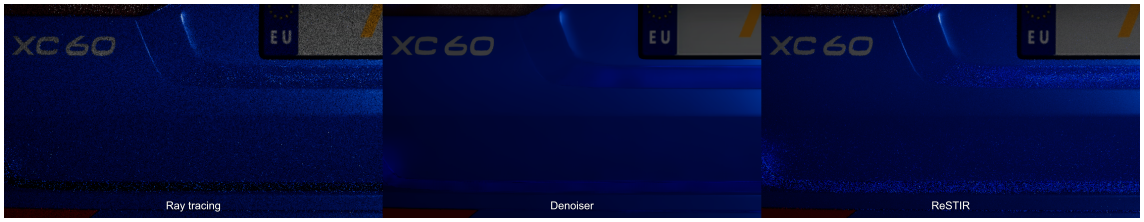


Figure 5.20: Denoiser in detail. Left side is traditional ray tracing, middle is applied the denoiser and right side is using ReSTIR

Figure 5.21 presents a comparative analysis of the computational cost of each method, depicted as a box plot of frame generation times over a 300 frame sequence. The baseline ray tracing serves as a performance benchmark, exhibiting a median frame time of approximately 10 ms with high consistency. The introduction of the denoiser as a post-processing stage increases the median frame time to roughly 12.5 ms. However, this method also displays the largest interquartile range and the greatest overall variance, as indicated by the box in Figure 5.21. This suggests that the computational cost of the denoiser fluctuates significantly from frame to frame, leading to less stable performance.

In contrast, the implementation of ReSTIR results in the highest computational overhead, with a median frame time of approximately 17.8 ms, representing a nearly 80% increase over the baseline. Despite this substantial increase in rendering time, the

frame time distribution for ReSTIR is considerably more stable than that of the denoiser. Its interquartile range is much narrower and is comparable to the consistency of the baseline ray tracing. This indicates a more predictable and uniform performance profile. Therefore, a clear trade-off emerges: while both techniques enhance visual fidelity at the expense of performance, ReSTIR achieves superior noise reduction and greater frame time stability, albeit at a higher computational cost than the denoiser.

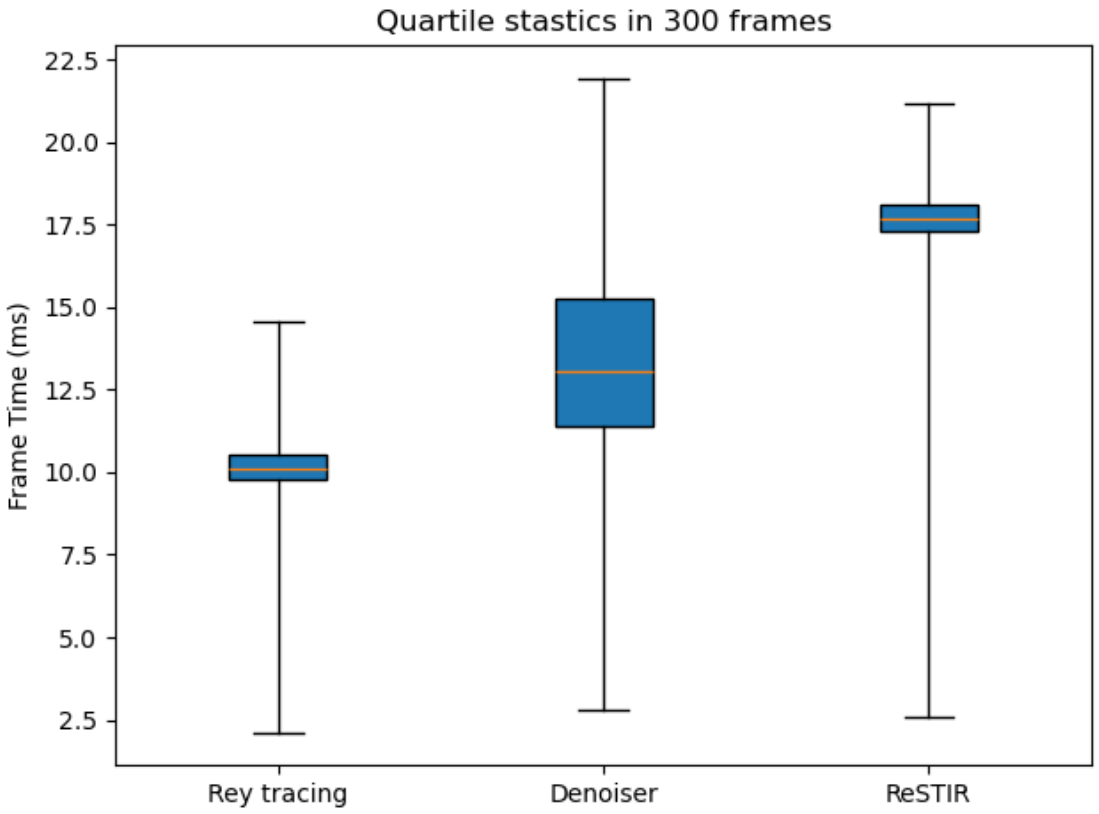


Figure 5.21: Box plots for frame time. Orange line is the median number of the frame time.

5.4 Latency of the real-time rendering

The integration of post-processing algorithms for image quality enhancement introduces critical performance trade-offs that demand rigorous analysis. Modern real-time rendering systems operate under strict temporal constraints, typically requiring end-to-end pipeline latency below 20 milliseconds to maintain perceptual continuity in interactive applications [43]. This requirement becomes particularly acute in automotive visualization systems where delayed feedback can compromise user experience and safety. Latency measurements were captured using Unity's built-in frame debugger, GPU profiling tools, and third-party diagnostic utilities. The focus was on identifying both frame-level latency (average milliseconds per frame), which influence perceived smoothness, and simulation stability.

When evaluating rendering performance, the 30 frames-per-second (FPS) benchmark (33.33 ms/frame) serves as the conventional threshold for perceived smoothness in human visual perception [44]. Experimental analysis reveals that unoptimized light pursuit implementations introduce substantial pipeline delays, significantly exceeding this baseline. The latency values illustrated in Figure 5.22 represent the worst-case measurements observed during the experiments.

Using Unity's default High Definition Render Pipeline, latency was consistently low, with frame times averaging around 9 ms. This configuration leverages deferred shading, screen-space effects, and GPU-efficient lighting models. The frame time remained stable across standard test scenes like the Cornell Box and car model environments.

Enabling a custom ray tracing pipeline introduced a noticeable increase in rendering latency, with frame times rising to 20 ms. In automotive-scale environments, especially when simulating high-resolution car models with detailed BRDFs, ray tracing latency increased further. The frame delay may reach to 50ms. Such delays, while acceptable for offline or semi-interactive reviews, may be less suitable for real-time driving simulator.

The introduction of the ReSTIR algorithm into the lighting pass further increased rendering latency due to the temporal and spatial sample reuse calculations. The highest delay when using ReSTIR reached up to 100ms. Although this technique reduces noise, the degradation of simulator availability caused by latency affects the user much more than the lack of image quality.

Notably, despite AVxcelerate and Luciddrive's shortcomings in terms of the simulator's image quality, their frame rates were pretty stable around 60 fps throughout the test. They were far more stable than Unity.

To explore the main sources of latency in Unity rendering, Unity's own analyzer is used. Through this analysis, it was observed that a significant portion of the frame time was consumed by the `WaitForSignal` function, which consistently appeared as the longest blocking call across multiple test scenarios. This function is part of Unitys low-level graphics job scheduling mechanism, responsible for synchronizing GPU and CPU execution. Specifically, `WaitForSignal` indicates that the CPU

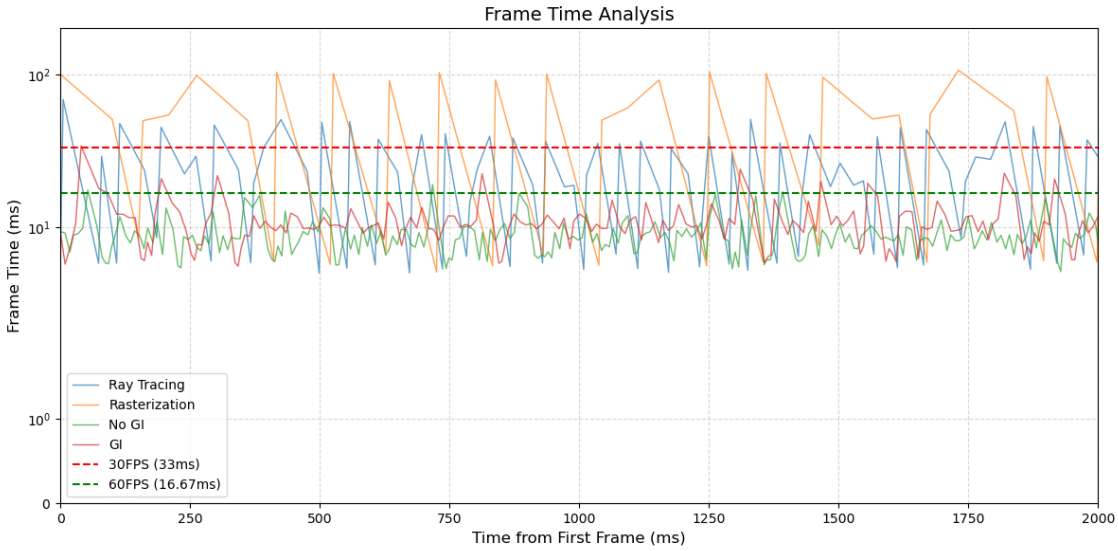


Figure 5.22: Frame latency in each ray tracing algorithms in Unity. Blue line and origin line is the the custom pipeline in Unity and green and red line is default Unity HDRP.

thread is stalling while waiting for the GPU to complete its previous frame rendering tasks. This suggests that further latency reductions may require optimizations at the shader level to reduce GPU operations per frame. The possible reason could be for logic errors in the code that are causing deadlocks or underutilization of the task process queue.

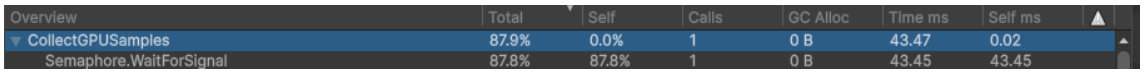


Figure 5.23: Main causes of delayed frame generation

This page was intentionally left blank.

6

Conclusion and Future Work

6.1 Conclusion

This thesis presented a comprehensive comparative study of light simulation software aimed at optimizing the fidelity of night driving simulations. By rigorously evaluating Unity enhanced with ReSTIR, Ansys AVxcelerate, and Synopsys LucidDrive, the research systematically analyzed the performance trade-offs between real-time and pre-rendered global illumination strategies. The evaluation was based on core metrics including BRDF realism, rendering architecture efficiency, photometric fidelity quantified by SSIM, and hardware utilization, using standardized test scenarios like the Cornell Box and automotive-grade models.

The results demonstrate that Unity's ray tracing pipeline, augmented by the ReSTIR algorithm, achieved the highest visual and photometric accuracy. ReSTIR significantly improved temporal stability and noise reduction, enabling high-quality global illumination. This approach excelled in simulating complex light-material interactions, such as indirect illumination and soft shadows, which are critical for validating adaptive headlamp systems. However, ReSTIR introduced minor spatial biases in geometrically complex regions, highlighting a trade-off between perceptual quality and absolute physical accuracy.

Ansys AVxcelerate delivered consistent photometric outputs, particularly in illumination continuity and sensor-range fidelity. Its strength lay in precise IES data interpretation, producing smooth light distribution maps essential for engineering validation. However, AVxcelerate relied on static baked shadows, limiting its ability to model dynamic lighting changes. Additionally, finer angular sampling resolutions improved edge sharpness but incurred higher GPU memory costs, underscoring scalability challenges.

Synopsys LucidDrive prioritized computational efficiency. While suitable for rapid iterations, its simplified illumination model struggled with advanced effects like diffuse inter-reflections and realistic BRDF variations. This make real time rendering indicating acceptable but not exceptional perceptual fidelity.

Key limitations of this work included the proprietary nature of the commercial software, which restricted granular control over rendering pipelines, and the fact that test scenes did not encompass extreme weather conditions. Furthermore, inconsistencies in IES file interpretation across platforms complicated direct photometric

comparisons. This thesis underscores that no single solution universally optimizes night driving simulations. The choice is use-case dependent: Unity with ReSTIR is preferred for design validation requiring high dynamic realism, AVxcelerate suits photometric engineering tasks, and LucidDrive offers efficiency for rapid prototyping.

6.2 Future Work

An emerging area of interest is the design of hybrid rendering systems that leverage the advantages of both real-time and offline strategies. For instance, dynamic elements of a scene could be rendered using high-performance real-time methods like ReSTIR, while static background elements are pre-rendered with computationally intensive path-tracing techniques to improve both realism and efficiency. This hybrid approach is particularly crucial for future large-scale, geometrically complex environments such as entire cities. In such scenarios, research must focus not only on the ray-tracing algorithms themselves but also on optimizing the underlying acceleration structures, such as Bounding Volume Hierarchies, to manage the immense computational load and enable real-time performance.

Furthermore, current validation metrics like SSIM, while valid for structural comparisons, do not fully account for human perceptual phenomena such as contrast sensitivity or motion artifacts. A recurring challenge is the lack of standardized methods and open-source resources for benchmarking rendering pipelines in automotive simulations. The establishment of a shared open-source dataset, including real-world measurement data and reproducible rendering configurations, would foster more consistent and transparent evaluations across different platforms.

Future work must therefore prioritize standardization, perceptual alignment, and adaptive rendering to achieve the dual goals of physical accuracy and real-time performance. By addressing these research directions, the automotive industry can accelerate the development of safer and more responsive lighting systems for the next generation of autonomous vehicles.

Bibliography

[1] C. Villa, R. Brémond, and J. Girard, "High dynamic range displays improve the realism of motion cues in night driving simulators," *Displays*, vol. 52, pp. 30–39, 2018, ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2018.02.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141938217301944.

[2] V. A. Frolov, A. G. Voloboy, S. V. Ershov, and V. A. Galaktionov, "Light transport in realistic rendering: State-of-the-art simulation methods," *Programming and Computer Software*, vol. 47, pp. 298–326, 2021.

[3] N. Rüddenklau, P. Biemelt, S. Henning, S. Gausemeier, and A. Trächtler, "Real-time lighting of high-definition headlamps for night driving simulation," *International Journal On Advances in Systems and Measurements*, vol. 12, no. 3 & 4, pp. 72–88, 2019.

[4] Ansys, *Autonomous vehicle simulation / avxcelerate headlamp*. [Online]. Available: https://www.ansys.com/content/dam/resource-center/datasheet/ansys-avxcelerate-headlamp-datasheet.pdf (visited on 05/09/2025).

[5] Synopsys, en. [Online]. Available: https://www.synopsys.com/optical-solutions/lucidshape/luciddrive.html (visited on 05/09/2025).

[6] Unity, en. [Online]. Available: https://unity.com/industry (visited on 05/09/2025).

[7] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[8] W. T. Neale, D. Hessel, and J. Marr, "Evaluation of photometric data files for use in headlamp light distribution," SAE Technical Paper, Tech. Rep., 2010.

[9] A. Keller *et al.*, "The iray light transport simulation and rendering system," in *ACM SIGGRAPH 2017 Talks*, Association for Computing Machinery, 2017, pp. 1–2.

[10] Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni, "Restir gi: Path resampling for real-time path tracing," in *Computer Graphics Forum*, Wiley Online Library, vol. 40, 2021, pp. 17–29.

[11] Cmglee, *File:Radiometry photometry units.svg - Wikimedia Commons — commons.wikimedia.org*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Radiometry_photometry_units.svg (visited on 05/20/2025).

[12] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters/crc Press, 2019.

[13] M. Xia, B. Walter, E. Michielssen, D. Bindel, and S. Marschner, "A wave optics based fiber scattering model," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[14] J. T. Kajiya, "The rendering equation," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 143–150, Aug. 1986, ISSN: 0097-8930. DOI: 10.1145/15886.15902. [Online]. Available: https://doi.org/10.1145/15886.15902.

[15] D. J. Eck, *Introduction to computer graphics*. David J. Eck, 2016.

[16] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016, ISBN: 0128006455.

[17] X. Li and A. L. Gilchrist, "Relative area and relative luminance combine to anchor surface lightness values," *Perception & Psychophysics*, vol. 61, pp. 771–785, 1999.

[18] I. R. Sector. "Bt.709 : Parameter values for the hdtv standards for production and international programme exchange." (2015), [Online]. Available: https://www.itu.int/rec/R-REC-BT.709 (visited on 05/09/2025).

[19] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Trans. Graph.*, vol. 1, no. 1, pp. 7–24, Jan. 1982, ISSN: 0730-0301. DOI: 10.1145/357290.357293. [Online]. Available: https://doi.org/10.1145/357290.357293.

[20] B. Burley and W. D. A. Studios, "Physically-based shading at disney," in *Acm Siggraph*, vol. 2012, vol. 2012, 2012, pp. 1–7.

[21] J. Dupuy and W. Jakob, "An adaptive parameterization for efficient material acquisition and rendering," *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 37, no. 6, 274:1–274:18, Nov. 2018. DOI: 10.1145/3272127.3275059.

[22] W. Jakob *et al.*, *Mitsuba 3 renderer*, version 3.1.1, https://mitsuba-renderer.org, 2022.

[23] E. Eisemann, M. Schwarz, U. Assarsson, and M. Wimmer, *Real-time shadows*. CRC Press, 2011.

[24] Z. Majercik, J.-P. Guertin, D. Nowrouzezahrai, and M. McGuire, "Dynamic diffuse global illumination with ray-traced irradiance fields," *Journal of Computer Graphics Techniques*, vol. 8, no. 2, 2019.

[25] M. Pharr and R. Fernando, *GPU Gems 2: Programming techniques for high-performance graphics and general-purpose computation (gpu gems)*. Addison-Wesley Professional, 2005.

[26] T. Zawistowski and P. Shah, *An Introduction to Sampling Theory*. [Online]. Available: http://www2.egr.uh.edu/~glover/applets/Sampling/Sampling.html (visited on 05/20/2025).

[27] C. Siegl, Q. Meyer, G. SuSSner, and M. Stamminger, "Solving aliasing from shading with selective shader supersampling," *Computers & Graphics*, vol. 37, no. 8, pp. 955–962, 2013, ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2013.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849313001271.

[28] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[29] M. A. Hassan and M. S. Bashraheel, "Color-based structural similarity image quality assessment," in *2017 8th International Conference on Information Technology (ICIT)*, IEEE, 2017, pp. 691–696.

[30] C. Lomont, *Problems using SSIM for image quality comparisons; Lomont.org — lomont.org*. [Online]. Available: https://lomont.org/posts/2023/ssim/ (visited on 05/20/2025).

[31] C. Wyman *et al.*, "A gentle introduction to restir: Path reuse in real-time," in *ACM SIGGRAPH 2023 Courses*, Los Angeles, California, Aug. 2023. DOI: 10.1145/3587423.3595511.

[32] 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.

[33] S. Zhang, D. Lin, C. Wyman, and C. Yuksel, "Many-light rendering using restir-sampled shadow maps," *Computer Graphics Forum*, vol. n/a, no. n/a, e70059, DOI: https://doi.org/10.1111/cgf.70059. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70059. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.70059.

[34] S. Niedenthal, "Learning from the cornell box," *Leonardo*, vol. 35, no. 3, pp. 249–254, Jun. 2002, ISSN: 0024-094X. DOI: 10.1162/002409402760105235. eprint: https://direct.mit.edu/leon/article-pdf/35/3/249/1571355/002409402760105235.pdf. [Online]. Available: https://doi.org/10.1162/002409402760105235.

[35] S. Lagarde and C. De Rousiers, "Moving frostbite to physically based rendering 3.0," *SIGGRAPH Course: Physically Based Shading in Theory and Practice*, vol. 3, 2014.

[36] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[37] Z. Hossein-Nejad and M. Nasri, "An adaptive image registration method based on sift features and ransac transform," *Computers & Electrical Engineering*, vol. 62, pp. 524–537, 2017.

[38] T. Saito and T. Takahashi, "Comprehensible rendering of 3-d shapes," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '90, Dallas, TX, USA: Association for Computing Machinery, 1990, pp. 197–206, ISBN: 0897913442. DOI: 10.1145/97879.97901. [Online]. Available: https://doi.org/10.1145/97879.97901.

[39] G. Lavoué and R. Mantiuk, "Quality assessment in computer graphics," in *Visual Signal Quality Assessment: Quality of Experience (QoE)*, Springer, 2014, pp. 243–286.

[40] *Cornell Box Data — graphics.cornell.edu*. [Online]. Available: https://www.graphics.cornell.edu/online/box/data.html (visited on 05/09/2025).

[41] J. Jimenez *et al.*, "Filtering approaches for real-time anti-aliasing.," *SIGGRAPH Courses*, vol. 2, no. 3, p. 4, 2011.

[42] L. Yang, S. Liu, and M. Salvi, "A survey of temporal antialiasing techniques," in *Computer graphics forum*, Wiley Online Library, vol. 39, 2020, pp. 607–621.

[43] S. LaValle, *The Latent Power of Prediction — developers.meta.com*. [Online]. Available: `https://developers.meta.com/horizon/blog/the-latent-power-of-prediction/` (visited on 05/20/2025).

[44] L. Salin, R. Morrar, L. Salin, and R. Morrar, "Game performance," *Game Development with MonoGame: Build a 2D Game Using Your Own Reusable and Performant Game Engine*, pp. 1–24, 2022.

A

Appendix 1

A.1 Each software function

	Real-time ray tracing	BRDF setting	IES file importing	Shadow
AVxcelerate	No	Yes	Yes	Not real-time
Luciddrive	No	Yes	Yes	Not real-time
Unity	Yes	Yes	Yes	Real-time

Table A.1: Function in each software

	Illumination view	Post-processing
AVxcelerate	Yes	Anti-aliasing
Luciddrive	Yes	No
Unity	No, need to write shader	Anti-aliasing, noise reduction

Table A.2: Function in each software (continue)