ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF AUTOMATION ENGINEERING

SECOND CYCLE DEGREE

……….

VIRTUAL PROTOTYPING FOR MACHINE VISION SYSTEMS

Dissertation in Automation Engineering

Supervisor

Professor Luigi Di Stefano

External Supervisor

Mr Luca Piccinini

Defended by

Zerun Hui

Graduation Session 21/07/2025

Academic Year 2024/2025

Abstract

KEY WORDS: MACHINE VISION, LED LUMINAIRE, VIRTUAL PROTOTYPING, BLENDER

This internship project was carried out at GENESI Elettronica, a company specialized in the design and manufacture of customizable electronic boards and lighting systems for the industrial department. The main goal of this project is to develop a software tool capable of virtualizing and simulating machine vision systems, focusing on lighting systems. The motivation of this research is based on the need to reduce production time and cost by eliminating the need for physical prototypes. The tool is designed to simulate the behavior of light from a source (illuminator) to an object and then to the camera, providing realistic image output that shows real-world conditions.

The research questions addressed in this project include: (1) How to develop a virtual prototyping tool to simulate complex vision systems? (2) What key functions does such a tool need so that it can accurately simulate lighting systems? (3) How to use the data indicators available in the simulation process to reflect the actual light intensity? (4) How to build a simulated lighting model in the software to generate high quality images?

The approach includes selecting a suitable development environment and framework, creating a virtual model of the LED luminaire based on the CAD design, and simulating light diffusion and intensity in 3D space. The main open source software used in this project to achieve the goal is Blender.

Key findings include the successful creation of virtual models of LED luminaires, the ability to simulate multiple LED modules within a designed structure, and the ability to adjust and visualize light behavior in a 3D environment. The developed software allows for the modification of various parameters and configurations, providing a flexible tool for testing different lighting setups. It is also able to simulate lighting systems in animations.

In summary, this project demonstrated the feasibility of using virtual prototypes to simulate machine vision systems, especially lighting systems. The developed

software not only reduces the need for physical prototypes, it also enhances the design process by allowing rapid testing and modification of system configurations. Future work may focus on expanding the functionality of the software, improving the accuracy of the observed data to include more complex vision system simulations and integrating other features to enhance usability.

Content

1. Introduction

1.1 Background

Machine vision is an essential technology in modern automation industry, which enables computers to interpret and analyze visual data in a manner similar to human vision. It integrates optical components, imaging devices, and software algorithms to execute tasks such as quality inspection, object recognition, measurement, and robotic navigation automatically. As the increasing demand for precision, efficiency, and scalability in the manufacturing industry continues to grow, machine vision has become a crucial tool of Industry 4.0, which promotes rapid development in industries such as automotive, electronics, pharmaceuticals, and logistics.

Considering the current state of machine vision industry, it has experienced rapid growth over the past decade, driven by advances in artificial intelligence (AI), deep learning, high-resolution imaging, and real-time processing capabilities. Traditional machine vision systems highly depends on rule-based image processing, while modern systems are increasingly using AI-driven techniques to improve accuracy and adaptability. These improvements mentioned before enable machine vision to handle complex, dynamic environments that were previously difficult to execute automatically.

Despite these advances, developing and optimizing machine vision systems is still a challenging process. Engineers normally need to build physical prototypes to test different lighting conditions, configurations of lens, and positions of camera. This approach is however:

◆Time-consuming, requiring multiple iterations to achieve optimal performance.

◆Expensive, as prototyping involves cost of hardware and setup.

◆Inflexible, The ability to test different environmental conditions will be limited if not modifying the physical setup.

To overcome these limitations, the industry is more and more turning to virtualization and simulation technologies. These tools provide a cost-effective, scalable, and efficient way for modelling, testing, and optimizing machine vision systems without relying on physical prototypes.

Virtualization and simulation allow engineers to design and improve machine vision systems in a virtual environment before implementation. Here are the advantages of these technologies:

◆Faster development cycles – Engineers can quickly test different configurations, lighting settings, and object interactions without physically assembling components.

◆ Lower costs – Virtual prototypes eliminate the need for multiple hardware iterations, saving resources, materials, and labor costs.

◆Improved accuracy – Advanced simulation tools can predict real-world performance by modeling light propagation, reflection, refraction, and object surface interactions.

◆Scalability – Systems can be tested in a variety of environmental conditions to simulate different possibilities, ensuring robustness before deployment.

◆Safety and feasibility testing – Engineers can evaluate system performance under extreme conditions (low light, glare, shadows, etc.) without exposing physical equipment to potential damage.

Several commercial and open source software solutions for optical and vision system simulation already exist, including:

◆Ansys SPEOS and Synopsys LightTools – widely used for industrial optical simulation, providing highly accurate light modeling.

◆Zemax OpticStudio and TracePro – specialized tools for lens and optical system design, providing in-depth analysis of light behavior.

◆Blender and WebGL – open source solutions that support 3D modeling and basic light simulation but may lack specialized vision system capabilities.

◆Unreal Engine – a gaming engine with advanced real-time rendering capabilities that can be used to simulate dynamic lighting conditions in industrial applications.

Although these existing solutions offer powerful modeling capabilities, they also have some limitations, such as high cost, complex interfaces, and limited customization options for specific industrial applications.

To look for solutions to these challenges, Genesi Elettronica aimed to develop specialized machine vision virtualization and simulation tools. The goal was to create software that would allow users to:

◆Design and configure vision systems (lighting, cameras, lenses, etc.).

◆Simulate real-world optical interactions with high accuracy.

◆Evaluate feasibility and optimize designs without physical prototypes.

◆Increase efficiency and reduce costs in machine vision system development.

This series of research and development is in line with Genesi Elettronica′s direction in innovation for industrial automation. Using advanced computer graphics, optical simulations and virtual prototyping, the company seeks to provide powerful, cost-effective and user-friendly solutions to meet modern manufacturing needs.

1.2 Research Objectives

The primary objective of this research is to develop an advanced software tool for the virtualization and simulation of machine vision systems, with a specific focus on lighting system modeling. This software tool will enable engineers to prototype, test, and refine vision system designs more efficiently.

The core research goals is to select best development environment. Evaluate software frameworks such as Blender, WebGL, Unreal Engine, and commercial CAD/simulation tools (e.g., Ansys, Zemax). Assess key factors such as rendering accuracy, ease of use, real-time performance, and computational efficiency. And then a realistic 3D LED model will be developed based on CAD designs provided by the company. Implement configurable light properties (wavelength, beam angle, intensity, etc.) to simulate real-world LED behavior. Additionally, IES photometric files, and allow to be imported for accurate optical performance modeling.

The simulation of Multi-LED Illuminators needs to be realized, enable the assembly of multiple LED modules onto a PCB structure to create complete illuminators. Allow users to modify LED positioning, beam configurations, and output characteristics dynamically.

3D Spatial Manipulation and Virtual Environment Interaction, which implement real-time navigation and manipulation of vision system components. Provide features for users to rotate, scale, and move illuminators within a 3D workspace. Allow placement of objects and lights at adjustable distances for realistic testing scenarios.

Incorporate high-fidelity physics-based rendering to model reflection, refraction, diffusion, and absorption of light, which ensure simulation outputs accurately predict real-world lighting conditions using spectral analysis techniques. An intuitive graphical interface allows users to input parameters and customize the system to adjust parameters such as LED spectrum, object geometry, ambient lighting conditions and object surface characteristics. This ensures flexibility for future enhancements and industry-specific applications.

Simulation of complete machine vision systems, supporting multi-component simulation and integration, such as illuminators (various LED configurations), cameras and optics (perspective/orthographic/panoramic camera models) and object surface and ambient lighting conditions. These allow users to test different configurations and analyze the combined effects of multiple system elements.

Here are the expected contributions and benefits:

◆Reduce R&D costs and time-to-market by minimizing physical prototyping.

◆Enhance accuracy in machine vision design through detailed simulations.

◆Facilitate rapid iterations and optimization in a virtual space before deployment.

◆Improve system robustness by testing under various environmental conditions.

◆Provide a specific tool for industry that meets the unique needs of manufacturing automation.

By achieving these objectives, Genesi Elettronica aims to revolutionize the machine vision system development process, making it more efficient, cost-minimize, and precise.

1.3 Thesis Structure

The following thesis is divided into several key sections to provide a comprehensive and structured introduction to the research conducted during the internship. These sections range from theoretical foundations to practical implementation and analysis.

Literature Review - This section examines existing research and technologies related to virtualization and simulation of machine vision systems. Find out how to implement these technologies in internship research.

Company Overview - This section introduces Genesi Elettronica's history, size, business scope, industry status, etc., and outlines its core business, expertise, and role in the development of industrial lighting and machine vision solutions. It also introduces the functions of the department and the specific responsibilities during the internship

Research Methodology - This chapter explains the research design and methods, including the selection of development tools, system architecture, and experimental steps, and explains data analysis techniques (quantitative analysis)

Internship Content and Process - This section details the activities carried out during the internship, describes the implementation process of the project in detail, and outlines the challenges encountered throughout the internship and their solutions. In addition, some further virtual lighting system construction processes are described.

Results and Analysis - This section presents the data or results collected during the internship. It evaluates the accuracy, efficiency, and usability of this tool and evaluates its potential impact on the design of machine vision systems.

Discussion – This section provides an in-depth analysis of the implications of the research findings. It discusses the strengths and limitations of the developed software, its practical applications in industrial automation, and potential areas for improvement.

Conclusion – The last section summarizes the main findings from the research and internship experience. It highlights the contributions of this research, its relevance to machine vision technology, and suggestions for future enhancements of the simulation tool.

2. Framework Review

With the rapid development of industrial automation and intelligent manufacturing, machine vision systems play an increasingly important role in quality inspection, robot navigation, product identification and other fields. Traditional machine vision system development often depends on the manufacture and field testing of physical prototypes. This method is not only time-consuming and labor-intensive, but also costly and difficult to adapt to the market's demand for rapid iteration and customized solutions. Therefore, virtualization and simulation technology have become the mainstream of development in recent years. By building digital prototypes in computers, researchers can simulate the working state of real optical systems in a virtual environment. As a result, the number of physical tests can be reduced, the development cycle can be shortened, and design accuracy can be improved.

In order to follow this trend, the company needs to test and analyze the performance of two rendering engines in Blender - Cycles and LuxCoreRender - to evaluate their effectiveness in simulating real-world lighting behavior. This comparative study is crucial to selecting the most appropriate virtual rendering tool for machine vision system development.

As an open source, cross-platform 3D modeling and rendering software, Blender has complete 3D content creation capabilities such as modeling, animation, materials, rendering, physical simulation, and video editing. It is widely used in film and television production, game development, industrial visualization, and other fields.

In the field of machine vision, Blender's biggest advantage lies in its powerful virtual modeling and realistic rendering capabilities. With built-in rendering engines (such as Cycles and Eevee) and support for external physical rendering engines (such as LuxCoreRender), Blender can be used to create a virtual simulation environment of real-world lighting, materials, and camera behavior. This environment is very suitable for training, testing, and verifying machine vision algorithms, especially when it is difficult to obtain a large amount of real image data or the cost of real-world testing is high.

Using Blender, researchers can:

- Simulate the impact of different light sources, materials, and backgrounds on the recognition effect of machine vision systems
- Build complex industrial scenes to test algorithm robustness
- Precisely control object positions, camera viewpoints, and lighting parameters for generating synthetic data sets

Therefore, Blender is becoming an important tool for building a machine vision virtual simulation platform, providing low-cost, high-efficiency data support and testing environment for scenarios such as intelligent manufacturing, industrial inspection, and robotic vision.

2.1 Cycles Engine

The Cycles engine in Blender is a global illumination renderer based on physical principles. Unlike traditional local lighting methods, The method of this engine pays more attention to the global propagation of light, so it can generate more natural and delicate lighting and shadow effects.

2.1.1 Monte Carlo Path Tracing

Cycles is Blender's physical rendering engine, which is based on the Monte Carlo path tracing algorithm. This is the most basic application theory. The application of Monte Carlo path tracing in optical simulation mainly achieves global illumination effects by simulating the random propagation path of light. The key of this algorithm is using probabilistic sampling methods to approximate the solution of complex Light Transport Equation (Kajiya, 1986).

$$\mathrm{L_{out}}(\mathrm{x}, \omega_{\mathrm{out}}) = \mathrm{L_e}(\mathrm{x}, \omega_{\mathrm{out}}) + \int \Omega \; \mathrm{f_r}(\omega_{\mathrm{in}}, \omega_{\mathrm{out}}, \mathrm{x}) \, \mathrm{L_{in}}(\mathrm{x}, \omega_{\mathrm{in}}) \, \cos\theta \; d\omega_{\mathrm{in}} \quad (1)$$

Where:

- $L_{out}(x, \omega_{out})$ is Outgoing Radiance: The total light energy leaving a surface point x in direction ω_{out}. It includes both emitted light and reflected/scattered light.

- $L_e(x, \omega_{out})$ is Emitted Radiance: The light directly emitted by the surface itself (e.g., a light source). If the surface is non-emissive, this term is zero.

- Integral Term

* Bidirectional Reflectance Distribution Function (BRDF): Defines how light from direction ω_{in} is reflected at point x toward ω_{out}. It encapsulates the material's reflective properties (e.g., diffuse, glossy, or specular).

* $L_{in}(x, \omega_{in})$ is Incoming Radiance: The light arriving at x from direction ω_{in} .

* $\cos\theta$ is Cosine Attenuation: Accounts for the projected solid angle, where θ is the angle between the incident light direction and the surface normal.
* $d\omega_{in}$ is Hemispherical Integration: Integrates over all possible incoming directions ω_{in} within the hemisphere Ω aligned with the surface normal.

The algorithm emits light from the camera or observation point and traces each collision and reflection process of the light in the scene until the light hits the light source or reaches a preset termination condition. For example, when rendering an indoor scene, the light from the camera may first hit the floor, and then randomly select a new propagation direction based on the reflective properties of the floor material - if the floor is a diffuse reflective material, the light will be randomly scattered according to the cosine distribution; if it is a specular material, it strictly follows the law of reflection. Each time the light interacts with the surface, the algorithm accumulates the lighting contribution on the path, including direct illumination from the light source and indirect illumination after multiple reflections and refraction. This process is achieved through recursive tracing, and finally the noise is reduced by statistical averaging of a large number of sampling results to generate a smooth image.

In practical applications, Monte Carlo path tracing needs to deal with a variety of complex optical phenomena. For example, for transparent materials such as glass or water, light may be reflected and refracted at the same time on the surface. At this time, the probability of the two paths needs to be dynamically allocated according to the Fresnel Equation. When simulating subsurface scattering, after the light enters the material, it will randomly change direction according to the phase function, gradually attenuate the energy, and finally emit from another position. This process needs to be achieved through volumetric path tracing. In addition, ambient light is usually used as an infinite light source through a high dynamic range (HDR) panorama. The algorithm needs to perform importance sampling on the panorama, giving priority to areas with high brightness (such as the sun or windows) to accelerate convergence. For participating media such as smoke and clouds, path tracing also needs to calculate the scattering events of light within the volume, randomly determine whether the light collides with the medium particles at each step, and accumulate the effects of absorption and scattering.

Although Monte Carlo path tracing has advantages in physical accuracy, its computational cost is high, especially when dealing with caustics, complex materials or low-light scenes, tens of thousands of samples are required to achieve visual noise-free effects. To this end, researchers have introduced a variety of optimization strategies. For example, Multiple Importance Sampling (MIS) reduces variance by combining light source sampling and material reflection direction sampling; photon mapping technology can pre-calculate the photon distribution of the caustic path and merge it with the path tracing results to accelerate caustic rendering. Real-time attempts rely on hardware acceleration and deep learning denoising, such as using GPUs to calculate millions of light paths in parallel, and then reconstructing smooth images at low sampling numbers through neural networks (such as NVIDIA OptiX Denoiser). These technologies have made Monte Carlo path tracing not only used for movie-level offline rendering, but also gradually penetrated into real-time game engines and virtual reality, becoming a key bridge connecting physical simulation and efficient rendering.

2.1.2 Multiple Importance Sampling

In the Blender Cycles rendering engine, Multiple Importance Sampling (MIS) is considered a key technology to optimize the sampling process, reduce noise, and accelerate image convergence. Traditional path tracing methods often require random sampling from many possible light paths, while in actual scenes, some light paths contribute much more to the final image than others. To this end, Cycles uses MIS technology to dynamically allocate sampling weights between different sampling strategies to achieve more efficient and accurate lighting estimation. Specifically, Cycles considers both direct sampling from the light source and sampling based on the material BRDF. It combines the probability density functions of the two, and merges the sampling results through the Balance Heuristic, so that even in the case of limited samples, it can ensure unbiased estimation and significantly reduce the noise caused by insufficient sampling of low-probability, high-contribution light paths.

The core of this method is that it can focus sampling on areas with greater contributions based on different light sources and material properties. For example, when dealing with highlight areas or caustics, direct light sampling may be more likely to capture key lighting information, while in other cases, it may rely more on BRDF-based sampling. Through Multiple Importance Sampling, Cycles can keep a good balance between the two sampling methods, so that the rendering process can avoid excessive noise caused by a single sampling strategy. And at the same time, it can also make full use of their respective advantages to improve overall rendering efficiency. This dynamic weight distribution mechanism of MIS ensures that throughout the sampling process, no matter which sampling method is applied, it can play a role in the final lighting calculation, making the image smoother and more realistic, and the convergence speed is also significantly improved.

Besides, Multiple Importance Sampling is also very flexible in practical applications. The Cycles engine allows users to enable or disable the MIS option according to specific circumstances in the material settings, which means that for materials containing self-illuminating shaders, the system can directly cast sampling rays to the light source instead of relying solely on the results of random bounces. This design is particularly effective in dealing with scenes where local lighting is weak but the overall ambient lighting is important, which helps reduce noise and optimize sampling distribution. In general, MIS technology not only improves the quality of rendering results, but also significantly improves the lighting calculation performance in real-time rendering by balancing the contributions of different sampling strategies while ensuring unbias. It is of great significance to the development of modern graphics rendering systems (Sadeghein, 2024).

2.1.3 Light Bounces

In the Blender Cycles engine, light bounce control is one of the core mechanisms of global illumination calculation. It allows users to set independent maximum bounce times for different types of light interactions (such as diffuse reflection, glossy reflection, transmission, and volume scattering) in order to balance the consumption of computing resources and the quality of the final image while ensuring the realism of the rendering. By limiting the number of light bounces, the system can effectively reduce excessive unnecessary sampling, therefore it can reduce noise and speeding up rendering, while avoiding the sharp increase in computing costs caused by infinite bounces.

Specifically, the Cycles light tree control function allows users to adjust multiple parameters in the rendering settings. For example, users can set the maximum number of diffuse bounces separately, which is usually because the contribution of diffuse light will decay rapidly after multiple bounces; similarly, for glossy reflection, since specular reflection often produces obvious highlights, more bounces may be needed to capture subtle changes, but increasing the number of bounces will also need more calculations. And for transmission (such as glass materials) and volume scattering (such as smoke or clouds), Cycles also allows the upper limit of bounces to be set separately, which can ensure that while keeping the transparency or scattering effect realistic, the rendering time will not be greatly extended due to too many bounces.

The advantage of this separate control is that technicians can make flexible adjustments according to the specific needs of the scene. For example, in an indoor scene, if the main focus is on the diffuse reflection effect of the wall and floor, the maximum number of diffuse bounces can be appropriately reduced to lower noise and improve rendering efficiency (Cline et al., 2005). While in scenes that need to show highlights and delicate reflections, the number of glossy reflection bounces can be appropriately increased to ensure that the details are fully displayed. At the same time, for transparent materials and volume effects, the number of bounces set independently can better control those complex light propagation phenomena, making the overall lighting more balanced and realistic.

What's more, the Cycles engine also cooperates with other parameters (such as light getting weaker, sampling number, and noise suppression strategy) in setting the number of bounces, which together determine the final rendering effect. If the number of bounces is too low, it may lead to insufficient indirect lighting in the image, showing a strong sense of flatness or insufficient lighting; while too many bounces will lead to more noise, longer calculation time, and even overexposure of details in some cases. Therefore, how to reasonably set the upper limit of the bounce of various types of light while maintaining visual realism and details has become an important optimization direction in real-time rendering.

In short, the light tree control function in Blender Cycles provides a flexible and efficient control method by allowing users to set independent bounce times for different light types. This mechanism not only helps to optimize the global illumination effect, reduce noise, and improve convergence speed, but also can be fine-tuned according to the scene requirements in practical applications, in order to find an ideal balance between realism and rendering efficiency. This design concept and implementation method have important practical significance in the field of modern computer graphics and provide valuable reference for other rendering systems.

2.1.4 Adaptive Sampling

Adaptive sampling is a technology used in Blender to dynamically adjust the number of sampling times according to the noise level of each area of the image, therefore the rendering efficiency can be improved and the unnecessary computation can be reduced. The basic idea of adaptive sampling is that during the rendering process, the system will evaluate the variance or noise level of each pixel or pixel block in real time. If a certain area has reached the preset convergence standard value (that is, the noise is low), the sampling of the area will be stopped. As for complex areas or high-contrast areas, the number of samples will be increased to further reduce the noise and ensure that the details are fully captured.

This technology is theoretically based on the unbiased principle of Monte Carlo integration (Pharr et al., 2002). By estimating the local statistical variance of each pixel area, statistical methods are used to determine whether it is stable enough or not. The key is to dynamically adjust the sampling rate to avoid wasting sampling resources on those areas that have converged, and concentrate more computing resources on those areas with drastic lighting changes or complex material characteristics. The literature points out that adaptive sampling can not only significantly reduce rendering time, but also reduce the noise caused by random sampling while ensuring the overall quality of the image, therefore improving the visual realism of the image.

In Blender Cycles, the implementation process of adaptive sampling usually involves the following steps: First, after each sampling cycle, the system will count the sampling results of the current pixel and calculate the noise level or variance index of the pixel. Then, it compares the index with the threshold set by the user; if the noise is lower than the threshold, the system considers that the pixel has reached the convergence state, and terminates the subsequent sampling of the pixel in advance; in the area with higher noise, the number of samples will continue to increase until the convergence requirement is met. This design not only enables the renderer to allocate computing resources more intelligently, but also effectively avoids too many unnecessary ray tracing calculations and reduces the overall rendering time.

In addition, adaptive sampling complements other optimization techniques (such as Multiple Importance Sampling, noise degradation, and light bounce control) to form the overall framework of Blender Cycles rendering optimization (Jensen, 2002). In high-quality rendering scenes, these technologies work together to not only ensure the realism and detail performance of the final image, but also make the rendering process more efficient and flexible. The literature review mentioned that the introduction of adaptive sampling has brought significant performance improvements to real-time rendering and provided strong support for further optimization of global illumination calculations in the future. Its theoretical and practical significance has been widely recognized in the field of modern computer graphics.

2.2 LuxCoreRender Engine

LuxCoreRender is an open source rendering engine based on physical principles. It not only supports traditional path tracing, but also implements a variety of advanced global illumination algorithms such as bidirectional path tracing, Metropolis light transport, and photon mapping, so that it can capture delicate light and shadow, caustics, and subsurface scattering effects in complex scenes (LuxCoreRender, n.d.).

2.2.1 Bidirectional Path Tracing

Bidirectional path tracing (BPT) is a global illumination algorithm that combines forward path tracing (starting from the camera) and reverse path tracing (starting from the light source), aiming to efficiently solve complex light path problems. The key idea is to generate light paths from both the camera and light source directions at the same time, and dynamically connect these paths in the scene to capture light transport paths that are difficult to sample with traditional methods. The mathematical basis of this method can be traced back to the bidirectional path tracing theory proposed by Veach and Guibas in 1995 (Veach & Guibas, 1995), and the basic formula is:

$$\mathrm{I} = \sum_{\mathrm{k}=1}^{\infty}\sum_{\mathrm{l}=1}^{\infty} \mathrm{W}_{\mathrm{k,l}} \cdot \Phi_{\mathrm{k,l}} \quad (2)$$

Where:

- $\mathrm{W}_{\mathrm{k,l}}$ is weight function
- $\Phi_{\mathrm{k,l}}$ is path contribution

In actual implementation, LuxCoreRender's BPT algorithm first emits a path from the camera (called the "camera subpath"), which records the interaction points between light and objects in the scene (such as reflection, refraction or scattering) one by one.

At the same time, another path (the "light source subpath") starts from the light source and records the propagation of light in the scene. Subsequently, the algorithm attempts to connect all possible camera subpath vertices with the light source subpath vertices to form a complete light transmission path. For example, when the third vertex of the camera subpath is connected to the second vertex of the light source subpath by a visible line segment, the contribution of the path will be calculated and accumulated into the final image. In this process, the Multiple Importance Sampling (MIS) technique is used to balance the weights of different path combinations to ensure unbiased estimation (Veach & Guibas, 1997). By dynamically adjusting the connection strategy, BPT can prioritize high-contribution paths (such as directly visible light sources or specular reflection areas), therefore it can significantly improve the ability to restore details at the same number of samples, especially when dealing with caustics or indirect diffuse reflections (Jensen, 2001).

LuxCoreRender's BPT also integrates other optimization techniques to meet the challenges of complex scenes. For example, when rendering volumetric effects such as smoke or fire, BPT combines the volume scattering model to calculate the absorption and scattering behavior of light in the medium, while capturing the propagation of indirect light in the medium through path connection. For multi-layer materials (such as coated metals or wet surfaces), BPT simulates the interaction of light between different material layers through an accurate BRDF (bidirectional reflectance distribution function) model to ensure energy conservation and physical accuracy (Pharr et al., 2016). In addition, LuxCoreRender introduces progressive rendering mechanism, in order to preview images in real time and dynamically adjust computing resources based on noise levels (Keller, 2012). For example, in the beginning rendering stage, the algorithm quickly generates low-sampled images and identifies high-variance areas (such as shadow edges or complex caustics), and then concentrates on increasing sampling in these areas, thereby balancing speed and quality overall.

Although BPT has significant theoretical advantages, its practical application still faces some challenges. First of all, the connection of bidirectional paths requires a lot of computing and memory resources, especially in the case of scene with high complexity or a large number of light bounces, which may lead to a significant increase in rendering time. In addition, the parameter tuning of BPT is relatively complex, and users need to have a deep understanding of light path depth, sampling strategy, and connection rules to fully realize its potential. Future development directions may include combining deep learning technology to predict high-contribution paths, optimizing heterogeneous computing (CPU and GPU collaboration) to accelerate path connection, and further simplifying the user interface to lower the technical threshold. Despite this, BPT has showed crucial value in scientific visualization, high-end product rendering, and film and television special effects, It has become an important tool for pursuing a balance between physical accuracy and artistic expression.

2.2.2 Metropolis Light Transport

The proposal and evolution of MLT have profoundly influenced the field of computer graphics. Since Eric Veach and Leonidas Guibas first proposed the algorithm in 1997 (Veach & Guibas, 1997), academia and industry have continuously optimized its implementation. For example, Kelemen et al. (Kelemen et al., 2002) simplified the complexity of the mutation operation by introducing the "Primary Sample Space", while recent studies have attempted to combine neural networks with MLT to guide path mutation strategies using generative models (Kelemen et al., 2002). As an open source rendering engine, LuxCoreRender has become an ideal platform for verifying these new methods with its algorithmic transparency and flexibility. In the field of scientific visualization, MLT has even been used to simulate the propagation of photons in complex media (such as biological tissues or optical fibers), showing its powerful potential.

The Metropolis Light Transport (MLT) used in the LuxCoreRender engine is an advanced global illumination algorithm based on the Markov Chain Monte Carlo (MCMC) method, which is specifically designed to solve the challenge of difficult efficient sampling of traditional path tracing in complex light paths. The key idea is to achieve low-noise rendering results with fewer samples by dynamically adjusting the light path generation strategy to gradually focus on areas that contribute significantly to the image. Traditional path tracing depends on completely independent random sampling, while MLT uses a "path mutation" strategy to make local modifications based on existing paths, such as slightly tunin the starting point, direction, or number of bounces of the light, to generate a series of progressively optimized new paths (Pharr et al., 2016). This mutation process not only retains the correlation between paths, but also effectively explores those high-contribution light paths that are easily ignored by traditional methods, such as caustic spots after multiple reflections or indirect illumination from hidden light sources.

In LuxCoreRender, the MLT workflow first generates an initial set of seed paths that quickly cover the main lighting patterns in the scene through traditional methods (such as path tracing). Subsequently, the algorithm enters an iterative phase to generate candidate paths through a variety of mutation operations (such as lens plane perturbations, light source direction adjustments, or path length extensions). The acceptance probability of each candidate path is determined by the ratio of its contribution value to the current path. If the new path is brighter, it is more likely to be retained; otherwise, it may be rejected. This mechanism ensures that the algorithm gradually concentrates on high-brightness areas while avoiding falling into local optimality. For example, when rendering the caustics of glass materials, MLT will prioritize exploring paths that form bright spots after multiple refractions, while traditional methods may require several times more samples to achieve similar effects due to insufficient randomness. In addition, LuxCoreRender dynamically adjusts the amplitude of mutations through an adaptive strategy: small and fine adjustments are made in high-brightness areas (such as caustic cores), while large jumps are made in low-contribution areas (such as uniform diffuse reflections) to quickly explore the space, therefore it can achieve a balance between efficiency and accuracy.

The advantages of MLT are particularly significant in complex scenes. Taking the rendering of transparent objects and volumetric media as an example, traditional path tracing often requires extremely high sampling numbers to cover all possible path variants when dealing with multiple refractions of light inside glass or scattering in smoke, while MLT can capture these complex light paths more intelligently through path mutations. For example, in a jewelry design scene, the complex refraction path inside the diamond is efficiently sampled through MLT's path extension mutation, which can reduce the rendering time to one-third compared to traditional methods while maintaining the integrity of details. Besides, MLT also performs well in indirect lighting, especially in closed spaces, where the soft shadows and color penetration effects formed by multiple diffuse reflections of light can achieve smooth transitions with a lower number of samples, significantly improving the artist's creative efficiency.

However, MLT also has some limitations. Its computational cost is large, especially when dealing with dynamic scenes or large-scale geometry, where path mutations and the calculation of acceptance probabilities may lead to a significant increase in rendering time. In addition, MLT is sensitive to parameter settings, such as the selection of mutation strategies and the control of path depth, which require certain experience, otherwise it may lead to slower convergence or biased results. LuxCoreRender partially solves this problem through multi-threaded parallelization and progressive rendering omit, allowing users to observe image progress in real time and dynamically adjust resource allocation, but the lack of hardware acceleration (especially GPU support) is still the main problem at present. Future research directions may include combining deep learning models to predict the distribution of high-contribution paths, or using heterogeneous computing architectures (such as CPU and GPU collaboration) to accelerate the mutation process, thereby further improving the ability of MLT.

All in all, LuxCoreRender's Metropolis Light Transport provides an efficient solution for difficult lighting scenes through its unique path optimization strategy. Despite the challenges of computational cost and tuning complexity, its outstanding performance in caustics, indirect lighting, and volume rendering makes it an important tool for pursuing a balance between physical accuracy and artistic expression. With the advancement of algorithm optimization and hardware technology, MLT is expected to open up new possibilities in real-time rendering, dynamic scenes, and interdisciplinary applications, and continue to push the boundaries of light simulation technology.

2.2.3 Photon Mapping

The photon mapping function in LuxCoreRender is a physically based global illumination solution method. It is usually used as an offline or hybrid global illumination algorithm in LuxCoreRender, which realizes efficient simulation of complex lighting phenomena (such as caustics, indirect lighting and specular reflection) in a two-step approach.

First of all, the photon mapping process is divided into two stages: photon tracing stage and radiation estimation stage. In the photon tracing stage, the system emits a large number of photons from the light source, which propagate, reflect, refract and scatter in the scene obeying physical laws. In this process, each photon that interacts with the surface of the object will be recorded and stored in a special data structure, and acceleration structures such as kd-tree are usually used to manage these photon data. The goal of this stage is to capture the distribution information of photons in the scene, including caustics and indirect lighting formed after multiple bounces (Hachisuka & Jensen, 2009).

Then, in the radiation estimation stage, when the renderer generates the final image, for each surface point, it queries its neighboring photons first. And next, it performs a weighted average of these photons based on factors such as distance, normal, and surface characteristics in order to estimate the radiation brightness of the point. This process actually converts the global illumination integral into a statistical estimate of the local photon distribution, which greatly reduces the number of samples required to directly solve the rendering equation and effectively reduces noise (Hachisuka & Jensen, 2009). Photon mapping is particularly suitable for capturing lighting effects that are difficult to sample directly through path tracing, such as complex caustics and indirect lighting caused by transparent materials.

When implementing photon mapping, LuxCoreRender not only follows the basic idea of traditional photon mapping, but also makes multiple optimizations in sampling strategy, data storage, and radiation estimation. For example, LuxCoreRender uses adaptive sampling technology to dynamically adjust the number of photon emissions according to the complexity of the scene during the photon tracing stage (Hachisuka & Jensen, 2009). And at the same time, in the radiation estimation stage, by using an efficient kd-tree algorithm, photons near the target surface can be quickly found, therefore ensuring rapid convergence in high-noise areas. In addition, LuxCoreRender may also combine photon mapping with other global illumination algorithms (such as path tracing or Metropolis Light Transport) to achieve a more robust and efficient hybrid rendering strategy, further improving the rendering effect in various complex scenes (Pantaleoni, 2017).

In general, the photon mapping function in LuxCoreRender achieves realistic reproduction of lighting phenomena such as caustics, indirect lighting, and complex material reflections by accurately simulating the propagation and interaction of photons in the scene. This technology can not only obtain high-quality images with less sampling, but also show great advantages in processing lighting paths that are difficult to sample directly. For visual effects production that require extremely high realism, photon mapping provides an effective global illumination solution, and also provides a valuable theoretical and practical foundation for further exploration of physically based rendering methods.

2.3 IES Files

The IES (Illuminating Engineering Society) file used in Blender is an industry-standard light source data format that is mainly used to accurately describe the light intensity distribution characteristics of lamps. This type of file can realistically simulate the light emission modes of various lamps in the real world, such as the focusing range of spotlights, the scattering effect of downlights, or the complex light patterns of car lights, by recording the light intensity values of light sources at different spherical angles (such as azimuth and elevation). The history of IES files can be traced back to the 1970s. Its standardized format (such as IESNA LM-63) was developed by the American Illuminating Engineering Society to provide a unified data interface for architectural lighting design, film and television lighting, and computer graphics (Walsh, 1959). In Blender, the application of IES files enables the lighting design of virtual scenes to be closely connected with the optical performance of the physical world, especially for architectural visualization, product prototype verification, or film and television special effects production that require high-fidelity rendering.

The key structure of IES files is divided into three main parts. The first part usually contains metadata about the lamp, which records information such as the model, manufacturer, product description, and installation method of the lamp. Although this information does not play a direct role in rendering calculations, it can help users identify the type of light source used and provides background support for subsequent lighting data interpretation. The second part is the measurement test information, which includes the number of bulbs in the luminaire, the luminous flux of each bulb, the number of angles used in the measurement process (usually including vertical and horizontal angles), and the geometric dimensions of the luminaire (IESNA Testing Procedures Committee, 2008). This part of the data is crucial for the correct interpretation of the subsequent photometric distribution information, because it directly determines the sampling density and distribution range of the photometric data. The third part is the key measurement data part, which records the light intensity values measured on a predetermined angle grid, which is the photometric grid. They are usually expressed in candela (cd) (IESNA Testing Procedures Committee, 2008). The measurement data is usually arranged in one or more rows of values, each of which corresponds to the light intensity at a specific angle combination. With this data, the renderer can reconstruct the luminous distribution of the light source in the virtual scene, the so-called "photometric solid".

In the process of applying IES files in Blender, it is usually necessary to import the IES file first, and then convert the photometric data in it into a format that Blender can recognize through a parsing tool or plug-in. A common approach is to use a dedicated IES file parser to map the measured data to a three-dimensional grid or texture map, and then associate this photometric texture with a point light source or an area light source through a node system. In this way, the virtual light source can present a non-uniform lighting distribution according to the actual measured data. For example, for a spotlight using IES data, its luminous angle and intensity are no longer simply uniformly distributed, but emit stronger or weaker light in a specific direction according to the angle data recorded in the file, thereby simulating the optical properties of the actual lamp.

Specifically, when the IES file function is enabled in Blender's material or light settings, the system will load the IES data into a dedicated texture node, which will generate a photometric map that not only contains light intensity information, but also reflects the luminous properties of the light source in all directions. When rendering, this photometric map will work together with other parameters of the light source (such as intensity, color, attenuation), and then finally affecting the propagation of light in the scene. In this way, IES files can make the lighting effects in virtual scenes closer to the performance of real lamps. For example, when rendering lighting scenes in buildings, IES files can be used to accurately simulate the beam distribution of lamps, edge softening effects, and brightness changes caused by light attenuation, therefore generating highly realistic images.

Through the above technical means, Blender can use IES files to achieve a high degree of restoration and precise control of the lighting characteristics of real light sources, which is of great significance for improving the accuracy of global illumination simulation and rendering effects. Photometric mapping based on IES data can not only improve the projection effect of light in the scene, but also provide detailed data support for subsequent lighting optimization and parameter adjustment, making the entire rendering process more physically realistic and flexible. Such technical applications undoubtedly provide a solid theoretical foundation and practical cases for the construction of virtual simulation platforms, and promote the popularization and development of digital lighting technology in various fields.

However, the application of IES files also faces some technical challenges. First of all, complex IES files may contain tens of thousands of light intensity values, which greatly increases the amount of calculations required during rendering, especially in global illumination scenes that require multiple light bounces. Secondly, the traditional IES format only supports static light intensity distribution and cannot simulate light sources with dynamic dimming or color changes, which limits its application in interactive content such as games or real-time visualization. Besides, some old versions of IES files or non-standard formats may cause parsing errors, and users need to use third-party tools (such as IES2LDT) for format conversion. These challenges are being addressed in a variety of ways. For example, NVIDIA's RTXGI technology combines ray tracing hardware acceleration to achieve real-time IES lighting rendering in dynamic scenes. Plugins developed by the Blender community (such as LuxCoreRender) further optimize the indirect lighting calculation of IES light sources and reduce noise generation through bidirectional path tracing.

In general, IES files, as a standardized photometric data storage format, record in detail the light intensity distribution of lamps at various angles, and become an important data source for realizing physically based global illumination calculations (Kajiya, 1986). In Blender, the application of IES files enables the renderer to accurately reproduce the lighting characteristics of the real world, so that the light sources in the virtual scene show complex and realistic beam distribution and brightness changes. Through the effective analysis and mapping of IES data, designers can use Blender's node system to easily build a light source model that conforms to the actual measurement data, therefore greatly improving the realism and visual effects of the rendered image. This technology not only provides a theoretical basis for virtual simulation and visual presentation, but also provides practical support for lighting design and energy efficiency optimization in actual industrial applications. In the future, with the continuous advancement of photometric data acquisition technology and rendering algorithms, the role of IES files in digital lighting simulation will become increasingly important, and its application prospects in architecture, industry, film and television, and virtual reality will also be broader.

2.4 Implementation Method

When using Cycles and LuxCoreRender engines in Blender for lighting simulation, the lighting effects of the real world can be accurately reproduced, but there is no method in the existing technology to directly export the lighting intensity. Both rendering engines provide a physically based global illumination calculation method. By using different ray tracing algorithms, they can simulate complex lighting interactions such as light emission, reflection, refraction, scattering, and light propagation in the medium. Cycles uses path tracing and bidirectional path tracing algorithms, combined with Multiple Importance Sampling (MIS) to optimize lighting sampling, ensuring high accuracy and efficiency when dealing with complex lighting environments; while LuxCoreRender introduces advanced lighting calculation technologies such as Photon Mapping and Metropolis Light Transport (MLT), further improving the rendering quality, especially when simulating caustics and complex indirect lighting. Although both engines can simulate real lighting accurately, they cannot output common light intensity indicators such as brightness and illumination through rendering results.

In this study, in order to achieve accurate simulation and quantitative measurement of the real lighting environment, we first used Cycles and LuxCoreRender engines in Blender to simulate real lighting, and then we needed to introduce the concept of pixel value. In computer vision and rendering, there is a direct relationship between pixel value and light intensity. Pixel value usually represents the color or brightness presented by each pixel in the image, while light intensity is the lighting intensity of a specific point or area in the scene. In the rendering process, light intensity determines how the area affects the brightness and color of the final image, which is usually measured by radiance. Radiance is the energy of light emitted from a point per unit area, usually quantified by direction and wavelength. When the rendering engine calculates the image, it determines the light intensity of each point based on the position, the intensity of the light source in the scene and the material properties of the surface of the object, and finally converts these intensities into pixel values. Thus, the final rendered image that conforms to the laws of physics is generated.

In general, although the rendering technologies used by Cycles and LuxCoreRender differ in the specific algorithm implementation, they both follow the physically based global illumination principle and use the Monte Carlo integration method to sample and estimate light transmission, which provides solid theoretical and practical support for the objectives of this study.. The pixel value is a direct representation of digital light intensity. Through reasonable measurement and conversion, quantitative evaluation of scene lighting can be achieved in the rendering system, and data support can be provided for subsequent image processing and optimization.

3. Company Overview

3.1 Description of Genesi Elettronica

Genesi Elettronica is a high-tech enterprise that established in Italy, which are mainly focusing on the research and development, design and manufacturing of customized electronic boards and industrial lighting systems. As an enterprise with strong innovation capabilities and technological skills, Genesi Elettronica has been committed to providing industrial customers with one-stop solutions from design to production since its establishment. Its products not only cover electronic control boards and LED lighting systems, but also extend to supporting optical testing and quality inspection services. Depending on a strict R&D system and advanced manufacturing technology, the company has gradually built a good brand image and industry reputation in the field of industrial electronics and lighting.

In terms of company history, Genesi Elettronica was founded many years ago and initially started with small-scale customized electronic products. With the continuous expansion of market demand, the company has gradually expanded from a single product line to a diversified business field. In recent years, with the rapid development of global industrial automation and intelligent manufacturing, the company has kept developing with the times and increased its investment in R&D and technological innovation. Especially in digital transformation and intelligent manufacturing, Genesi Elettronica has achieved high integration and intelligent control of production processes by introducing advanced information management systems and automated production equipment. In 2017, the company completed a large-scale expansion in the Spilamberto area, expanding the total area of its production and R&D base to approximately 3,500 square meters. It now has product development laboratories, production departments and a complete office, administrative and technical support system. This series of measures has not only greatly improved the company's production capacity, but also laid a solid foundation for it to seize the initiative in the highly competitive industrial market.

In terms of business scope, Genesi Elettronica is committed to providing customized and modular electronic board and lighting system solutions for different industries. With commercial software and advanced CAD/CAM technology, the company is able to design unique electronic control systems and energy-efficient lighting products according to the specific needs of customers. Whether it is industrial automation, production line inspection, or high-precision visual systems, Genesi Elettronica can provide a complete set of services from concept design, sample development to mass production. In addition, as the market's requirements for green environmental protection and energy efficiency increase, the company is also constantly introducing intelligent control and energy consumption management technologies in LED lighting products, trying to reduce energy consumption while ensuring high performance and achieving sustainable development.

In terms of industry status, Genesi Elettronica has won the trust of many well-known domestic and foreign companies with its outstanding technical strength and strict quality management system. The company passed the ISO 9001:2015 quality management system certification in 2020, which not only indicates that it has reached the international advanced level in quality control and process flow, but also lays a solid foundation for the company to further expand the international market. As a technology-driven enterprise, Genesi Elettronica has obvious advantages in customized solutions, rapid response to customer needs and efficient integrated design, which enables it to maintain a leading position in the fierce market competition. At the same time, the company also pays attention to establishing strategic cooperation with universities, scientific research institutions and other technology companies, and continuously improves its R&D capabilities and market competitiveness through the integration of industry, academia and research.

From the perspective of strategic development, Genesi Elettronica actively explores the application of new materials, new processes and new technologies, and continuously optimizes product structure and production processes to meet the ever-changing market needs and technical challenges. While improving product quality, the company also focuses on the research and development of new intelligent lighting and automation control technologies, trying to occupy a more important position in the future field of intelligent manufacturing and industrial Internet. In addition, the company has gradually expanded its market focus to internationalization, and are trying to create a globally competitive brand image by participating in international exhibitions, establishing overseas offices and strengthening cooperation with foreign customers.

Overall, Genesi Elettronica has established a good market reputation and brand image in the field of industrial electronics and lighting systems with years of technical accumulation and continuous innovation capabilities. The company not only has a complete R&D, production and quality control system, but also leads the industry in customized solutions, intelligent manufacturing and green environmental protection. In the future, with the acceleration of the global industrial automation process and the growing demand for intelligent manufacturing, Genesi Elettronica is expected to demonstrate stronger competitiveness and development potential on a broader international stage with its deep technical background and flexible market strategy.

By continuously strengthening technology research and development, improving production systems and actively expanding international markets, Genesi Elettronica is leading the development direction of industrial electronics and intelligent lighting systems. Its products and solutions not only meet the requirements of high-end industrial fields for precision control and high efficiency, but also provide flexible and customized solutions for small and medium-sized enterprises, promoting the advancement and innovation of technology in the entire industry. In the future, with the continuous innovation of technology and the continuous upgrading of market demand, Genesi Elettronica will continue to be committed to providing more efficient, intelligent and environmentally friendly products and services to global industrial customers, so as to achieve sustained and steady development in the fierce international competition.

3.2 Function of the Department and Responsibility during Internship

This internship project was carried out in Genesi Elettronica's R&D department. The main functions of Genesi Elettronica's R&D department include promoting the research and development of advanced technologies and applying them to the company's products and solutions, providing customers with customized electronic systems and intelligent control solutions. As the core force of the company's technological innovation, the R&D department focuses on multiple technical fields, especially in industrial automation, intelligent lighting, and machine vision systems. The work of this department covers the entire process from demand analysis, system design to prototype verification and product optimization, including hardware design, software development, optical simulation, data processing and integration. The R&D team works closely with other departments to ensure that the technical solutions can meet market demand and customer specific requirements. Through regular technical exchanges and innovative thinking collisions, the department has cultivated a group of technically skilled engineers and R&D personnel, and continuously promoted the company's technological progress and industry competitiveness. Overall, Genesi Elettronica's R&D department plays an important role in the company's technological innovation, continuously promoting the research and development and application of new technologies such as intelligent automation and precision control, and continuously providing industrial customers with more forward-looking and competitive solutions.

Regarding the responsibilities during the internship, since the virtualization and simulation technology of machine vision systems is one of the current key research and development directions, this internship project is committed to building a virtual prototype system based on physical principles, aiming to preview and verify the performance of machine vision systems in actual industrial applications through software simulation. During the implementation of the project, advanced computer graphics, optical simulation and data processing technologies are required to integrate CAD models, optical parameters and real-time rendering algorithms to establish a highly accurate and customizable virtual simulation platform. This platform can not only simulate the imaging effects under various lighting environments, but also export data such as light intensity to help engineers quickly determine the optimal solution in the early stages of design, thereby greatly reducing the cost and time of physical prototyping.

Specifically, the work during the entire internship covers multiple levels. First of all, for the lighting and imaging modules in the machine vision system, it is necessary to learn a variety of physical simulation algorithms, such as Monte Carlo integral-based path tracing, multiple importance sampling (MIS) and Metropolis Light Transport

(MLT) strategies. These technologies can realistically simulate the propagation, reflection and scattering of light in complex scenes in Blender, providing real global illumination calculations for machine vision systems. At the same time, special attention has been paid to the handling of advanced lighting phenomena such as volume rendering and subsurface scattering (SSS). These technologies can ensure high-quality visual effects when dealing with transparent materials, translucent materials, and scenes with complex caustics effects.

Secondly, in the research and development stage, we focus on system integration and optimization, and are committed to seamlessly connecting high-quality rendering technology with application scenarios such as real-time data acquisition and quality inspection. Through virtualization simulation technology, we hope to simulate machine vision workflows in various industrial environments in computers, and be able to measure the light intensity, reflection distribution, and color characteristics in virtual scenes. This can greatly help optimize the design of actual products in the future. This method not only greatly shortens the R&D cycle, but also improves the accuracy of the design and the market competitiveness of the product.

4. Research Methodology

4.1 Research Design

This study adopts a mixed methods research design, trying to build a bridge between theoretical analysis and practical application, in order to comprehensively explore and verify the application effect of machine vision system virtualization and simulation technology in industrial automation. The overall research framework is divided into three stages: concept construction, experimental design and data analysis. In the first stage, it is necessary to study the literature and online teaching videos, so as to better understand and master the basic principles and operation methods of Blender software, and then it is necessary to clarify the core issues, key variables and hypothetical models of the research. This stage includes both a deep analysis of existing virtual simulation technology and a preliminary understanding of the actual needs of industry, with the aim of providing theoretical support for subsequent experimental design.

In the second stage, the experimental design stage, this study will build a complete virtual simulation platform in the laboratory. The platform is based on mainstream rendering engines (such as Blender's Cycles and LuxCoreRender), and combined with advanced optical simulation algorithms and data acquisition systems to simulate the lighting conditions and operating conditions of machine vision systems in real industrial environments. At the same time, a physical model corresponding to the simulation environment will be built in the laboratory to record the real data of the actual physical model. During the experiment, control experiments with different parameter combinations (such as the working power of the light source, the distance between the light source and the reference object) are designed to observe the impact of various factors on the overall performance of the system. In addition, quantitative measurement tools are introduced. First of all, the pixel values of the final output rendering images of the key areas in each experimental scene should be systematically recorded and analyzed, and secondly, the illuminance of each key area on the reference surface of the actual physical model will be recorded.

The third stage mainly focused on data analysis and result verification. This study will mainly use quantitative data analysis methods to statistically analyze the changes in various experimental indicators. Through comparative analysis, the connection between the actual illuminance of the reference surface and the pixel value of the rendering output by Blender need to be found, and find the actual optimal parameter configuration and the optimal rendering strategy, in order to maximize the computational efficiency and rendering speed while ensuring the realism and detail performance of the image.

In terms of data collection, this study mainly relies on automated script technology in Blender and sensor monitoring technology in the actual physical model to compare and calibrate the illumination parameters in the virtual environment with the actual measured data. All experimental data have undergone rigorous preprocessing and statistical analysis to ensure the accuracy and reliability of the results. At the same time, in order to improve the depth and breadth of data analysis, it is necessary to visualize the data and present the key data indicators in the form of charts so that the research results can be seen at a glance.

Overall, the design and methods of this study have the following significant features: Firstly, the study adopts a hybrid method, focusing on theoretical modeling and literature review, and combining actual experimental data for quantitative analysis; secondly, the study uses a variety of advanced rendering and optical simulation technologies to construct realistic virtual scenes in the experimental design stage, and compares them with the data obtained from the actual physical model, therefore providing comprehensive technical verification for industrial practical applications. And finally, the study makes full use of statistical and visualization tools in data analysis to ensure that the results are highly reliable and repeatable.

Through this study, we hope to promote the application of virtual simulation technology in the fields of industrial electronics and intelligent manufacturing, realize the effective transformation from theory to practice, and bring more efficient, intelligent and low-cost technical solutions to the industry.

4.2 Data Collection Methods

This study mainly adopts a quantitative strategy in data collection, which can obtain comprehensive and accurate illumination data from the virtual simulation platform and the actual physical model, providing a solid data foundation for subsequent algorithm optimization and model verification. Specifically, the data collection method mainly includes two parts: virtual data collection and physical measurement data collection.

In the virtual data collection part, a high-fidelity virtual scene is first built using the Cycles and LuxCoreRender rendering engines in Blender. By writing an automated script, the system first renders and outputs the image of the reference surface, and then measures and outputs the pixel values of the key positions in the image. Each frame of the rendered output image is carefully preprocessed, including gamma correction, color correction and other steps to ensure that the pixel values in the image can truly reflect the intensity and distribution of the illumination in the scene.

In terms of physical data collection, a physical model corresponding to the virtual scene is built in the laboratory, and a light sensor is installed on the key reference surface. The sensor can record the actual illumination and radiation intensity of the area in real time, and these data are recorded as standard data in the real environment. In order to ensure the reliability of the data, the data at the same point on the reference surface will be measured multiple times before being recorded.

4.3 Data Analysis Techniques

In the data analysis stage, this study mainly uses quantitative analysis methods to conduct comprehensive and systematic statistical and comparative analysis of the collected virtual scene data and physical model data to verify the accuracy and reliability of the virtual simulation platform and explore the influences of different parameter configurations on the lighting simulation effect. Specifically, after collecting the data of the virtual scene and the real physical model, fill them into the table. For the pixel values of the reference surface obtained in the virtual scene, it is necessary to preprocess them first. This is because the value range of the pixel value is limited, and all the obtained pixel values must be within a reasonable range. The specific method will be described in detail in the next chapter. For the data recorded in the physical model of the laboratory, a preliminary comparison between the data is required to verify its reliability.

In order to more intuitively display the data analysis results, this study also uses data visualization tools to generate multiple scatter plots. These charts not only help to intuitively observe the lighting distribution of each key area under different parameter configurations, but also reveal the relationship and deviation between virtual data and physical data.

5. Internship Content and Process

5.1 Tasks, Responsibilities and Implementations

During my internship at Genesi Elettronica's R&D department, I participated in projects related to machine vision system virtualization and simulation technology. The entire internship process ran through various links from theoretical learning, software research and learning, system design, experimental verification to data analysis, and each stage involved some specific tasks and responsibilities. The following will describe in detail the main tasks, responsibilities and implementations I undertook during the internship, trying to fully reflect the work content of the R&D department in this field, and provide theoretical basis and practical support for subsequent technical improvements and research.

First of all, my first task was to deeply study and understand the concept of IES files and their role in lighting simulation. IES files are developed by the American Illuminating Engineering Society and a standard format for recording the photometric distribution data of actual lamps. The photometric distribution of lamps determines the distribution of their light intensity, and the distribution of light intensity will directly affect the final experimental results, so how to correctly apply IES files is very important. At the beginning of the internship, I consulted literature, online teaching videos and technical documents to understand the structure, content and measurement principles of IES files in detail, and mastered how to parse the lamp metadata, test parameters and photometric distribution data contained in the file. In order to better understand the influence of IES files on virtual light source effects, I also downloaded the IES file editor, so that IES files could be directly created through the editor and imported into Blender to observe its influence on the light source and understand how it changes the photometric distribution of lamps. Through this stage of learning, I not only had an intuitive understanding of the photometric characteristics of real light sources, but also laid a solid theoretical foundation for the subsequent construction of high-realistic light sources in virtual simulation platforms.

After understanding the basics of IES files, a virtual simulation platform has been built. The platform mainly relied on Blender's Cycles rendering engine and LuxCoreRender renderer, and its goal was to simulate the lighting conditions in real industrial environments and the operating status of machine vision systems. The specific tasks included importing product design models from the CAD system into Blender. These models included a PCB board, a background board to receive light, a light source of type 'Area', and a camera of type 'perspective'. Since not everything would have impact on the light effects, it was not necessary to build exactly the same virtual platform as the one built in the Lab. Then embedding the appropriate type of light source in the correct position in the model (Figures 1, 2 and 3), and creating an accurate photometric distribution model for the light source in the scene based on the IES file data (Blender Foundation, n.d.). In order to realize this, the IES data needed to be mapped to the lighting texture through the node editor (Figure 4) to ensure that the lighting effect in the virtual scene can be consistent with the actual lighting performance.

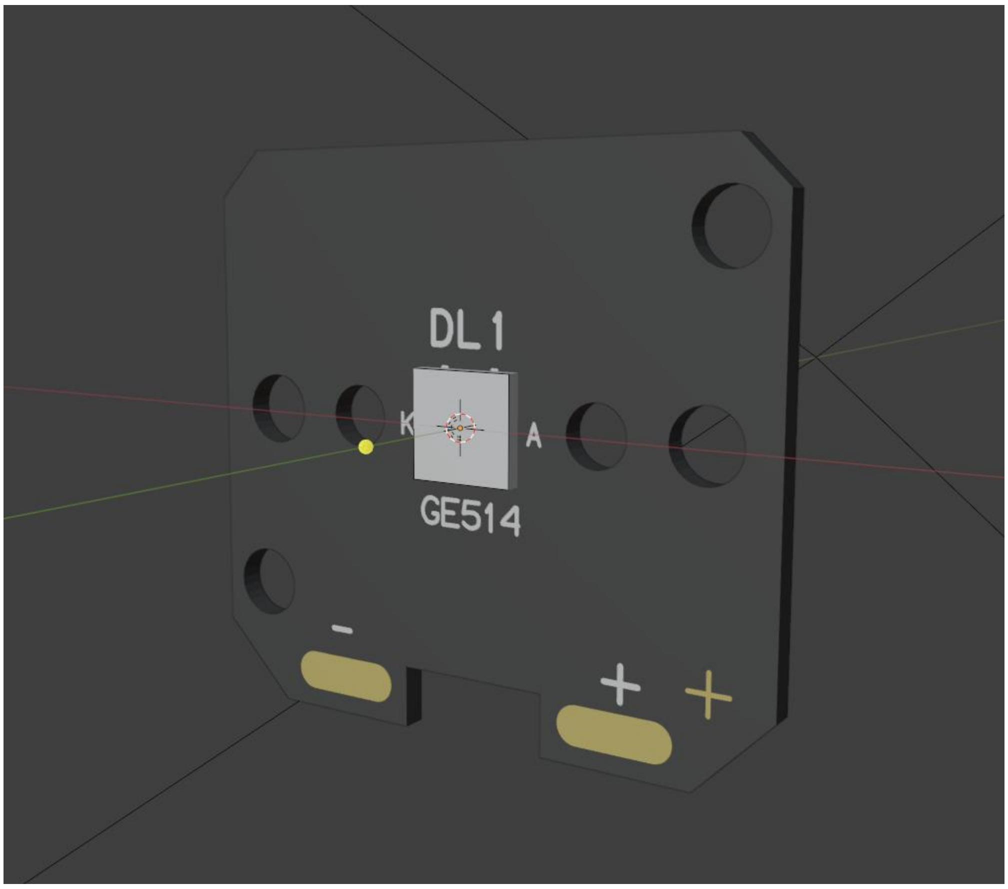


Figure 1. Embedding an Area Light Source in the Scene

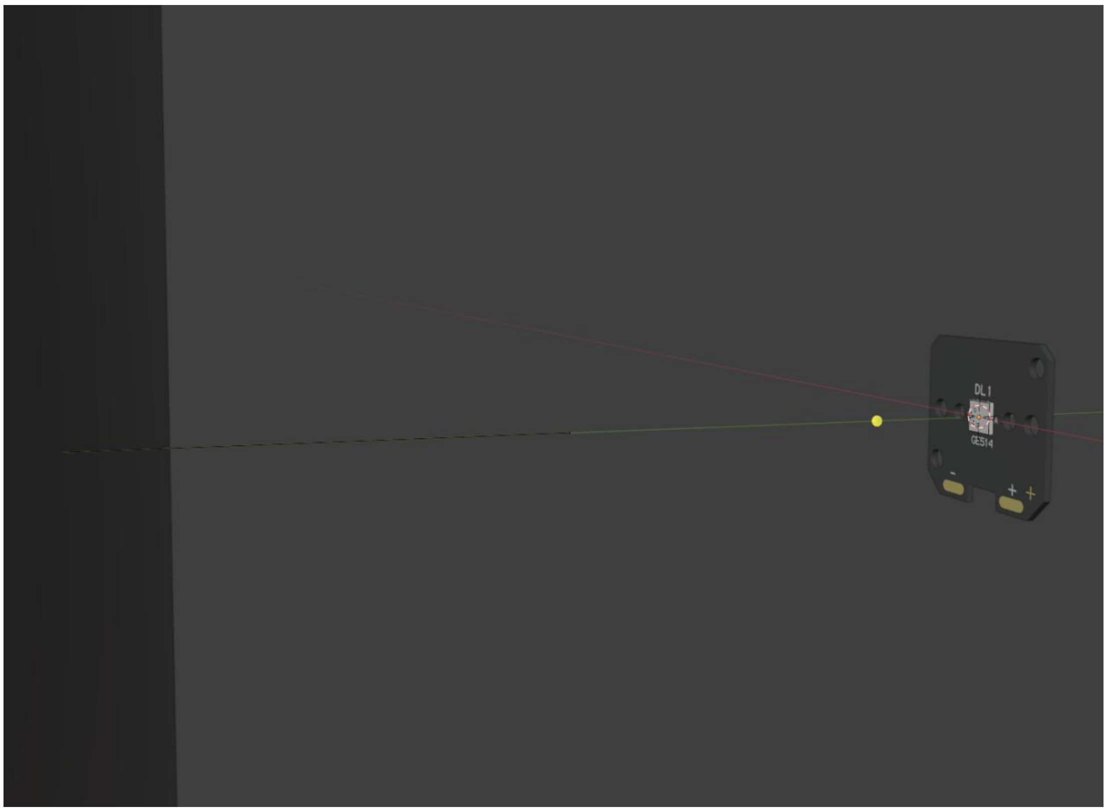


Figure 2. The position of Light Source in the Scene (a)

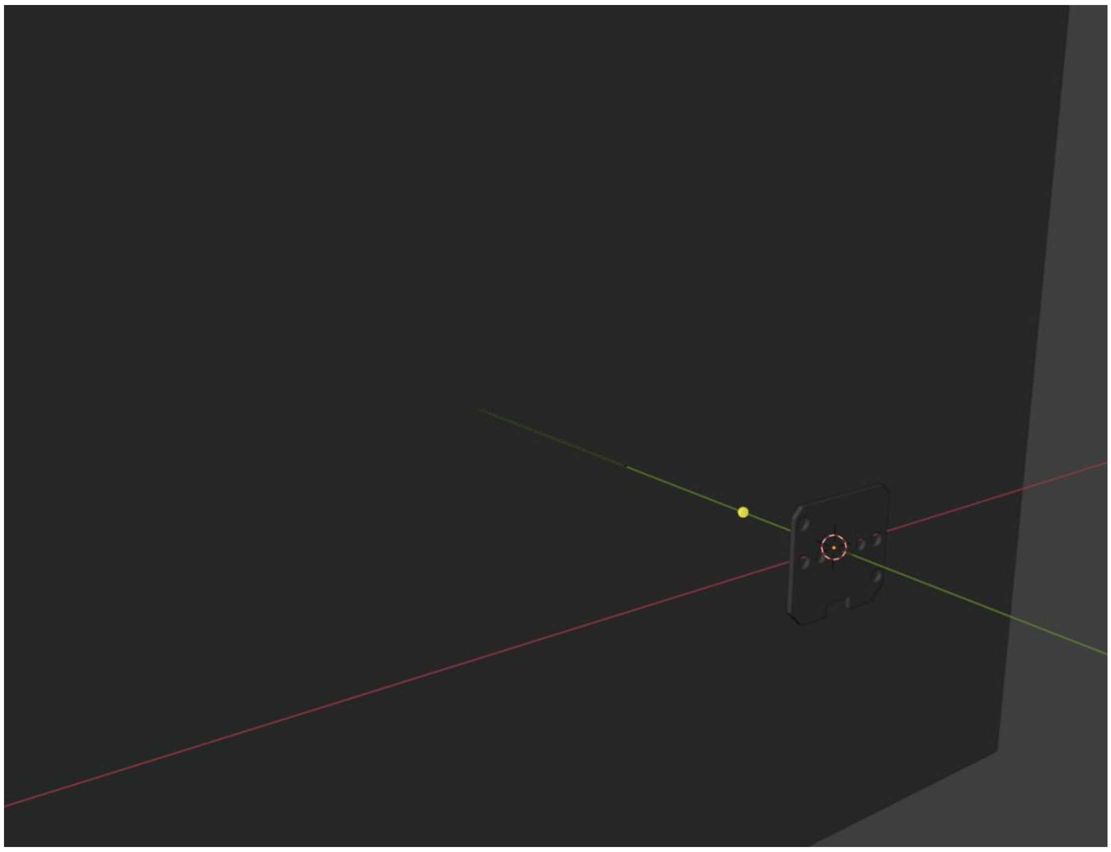


Figure 3. The position of Light Source in the Scene (b)

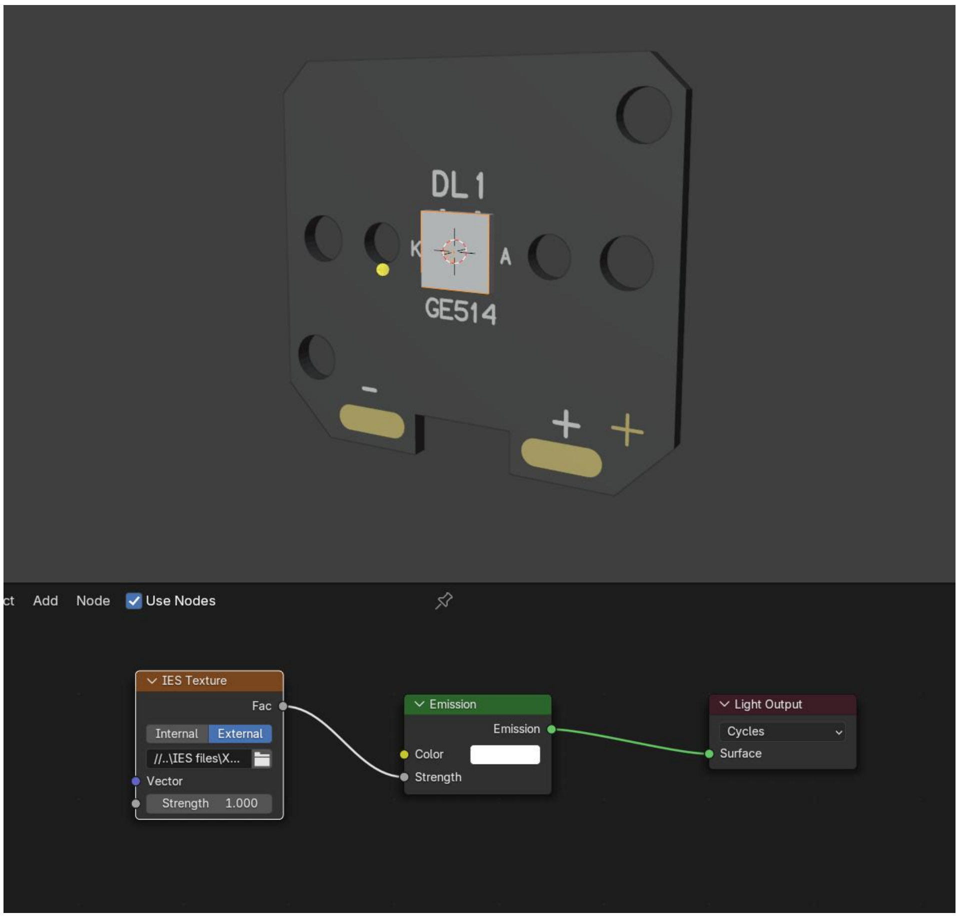


Figure 4. Mapping IES Data to Lighting Texture via the Node Editor

In order to ensure the accurate rendering of all devices in the virtual scene (such as light sources, cameras, reflectors, etc.), it was necessary to adjust the material of the background board used to receive light in the virtual scene and select the appropriate reflectivity, roughness, and refractive index (Shirley, 1991). Adjusting the position and angle of the light source so that the propagation path and focusing effect of light in the scene could accurately reflect the real optical behavior. Through repeated debugging and comparison, an ideal virtual model platform was provided for subsequent data analysis and system optimization (Gadia et al., 2024).

After the platform is built, it was necessary to further be responsible for experimental testing and data collection of virtual lighting effects. In order to simplify the control process, code could be used to achieve global control. The functions of the code are as following: firstly, set a function that can read the pixel value of the final rendered image; secondly, set the resolution of the final rendered image according to requirements; thirdly, set the final file format according to requirements; fourthly, select 'Cycles' or 'LuxCoreRender' as the rendering engine; fifthly, after selecting the corresponding rendering engine, set the rendering parameters of the currently selected engine according to requirements; finally, for the final output image, select the specified location where the pixel value needs to be read. In short, it is to use the built-in automated script function of Blender to sample each key area in the virtual scene and record the pixel values of each area. The full implementation of this function is provided in Appendix A (Code Implementation).

Data collection and comparative analysis are key links in this internship. For the final rendered image exported from Blender, its resolution is 800*600. In order to match the size of the real physical model background plate in the laboratory, it was placed in a 400*300 coordinate system (Figure 5). The pixel value was read every 30mm on the horizontal and vertical coordinates. In order to obtain the pixel value of the rendered image more accurately, the color depth of the image output was set to 16 bits instead of 8 bits, which could not only increase the value range of the pixel value, but also made the result more accurate by further subdividing the pixel value. This is because when the color depth is 8 bits, there are a total of 256 colors, while 16 bits has 65536 colors. After collecting the data of the virtual platform, charts were drawn, trying to find the relationship between distance and pixel value.

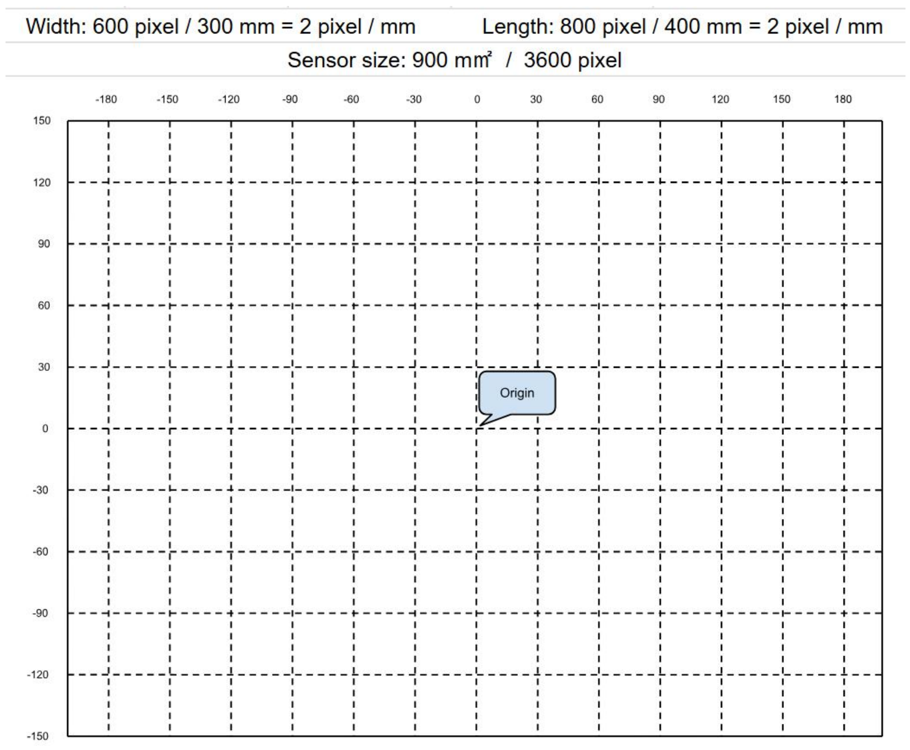

Width: 600 pixel / 300 mm = 2 pixel / mm
Length: 800 pixel / 400 mm = 2 pixel / mm
Sensor size: 900 mm² / 3600 pixel
-180
-150
-120
-90
-60
-30
0
30
60
90
120
150
180
150
120
90
60
30
0
-30
-60
-90
-120
-150
Origin

Figure 5. Placement of the Rendered Image in a 400×300 Coordinate System

For the actual physical model in the laboratory, the background plate range was set to 400mm*300mm, and the point where the center point of the light source is vertically projected onto the background plate was taken as the origin of the coordinate axis. Similarly, the illuminance and light intensity were read every 30mm on the horizontal and vertical coordinates (Figure 5). The reading tool included two sensors, an illuminance sensor and a light intensity sensor (Figure 6). In the laboratory, it was necessary to simulate the same environment as the virtual platform as much as possible to reduce the impact of ambient light on the experimental results. For the illuminance and light intensity at the same point, repeated sampling was used to improve the reliability of the data. After all the data was collected, charts were drawn, trying to find the relationship between distance and illuminance, distance and light intensity.

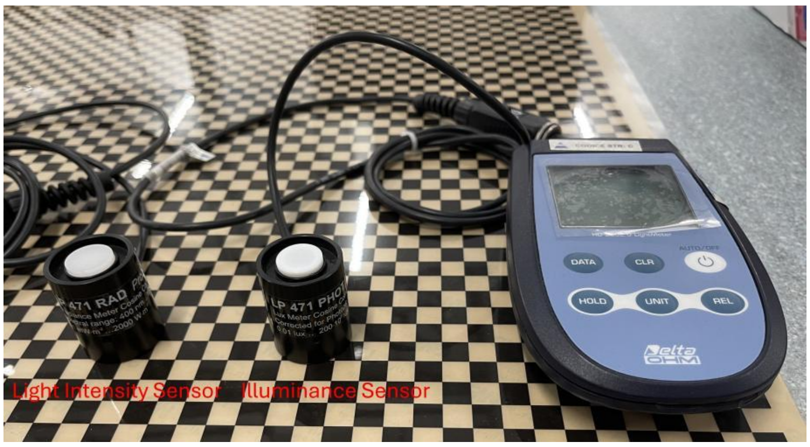


Figure 6. The Reading Tool with Illuminance and Light Intensity Sensors

During the entire internship, I also undertook some project reporting work. After completing each stage of work, I would record the design ideas, experimental setup (Figure 7), parameter settings (Figure 8) and test results of the project, and upload these contents to the online form. I also recorded the problems encountered in the experiment, listened to suggestions and found solutions through communication and feedback with the supervisor.

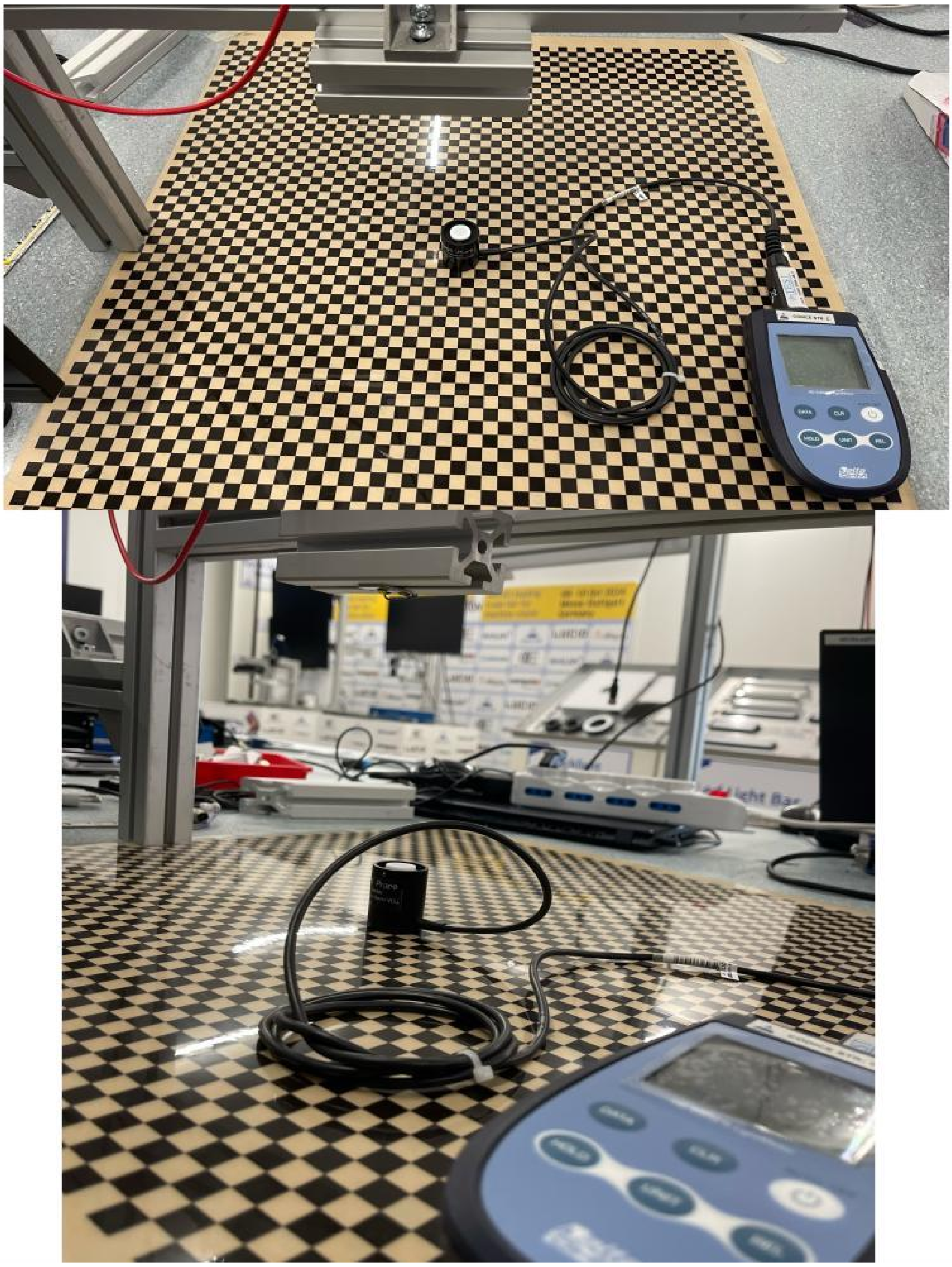
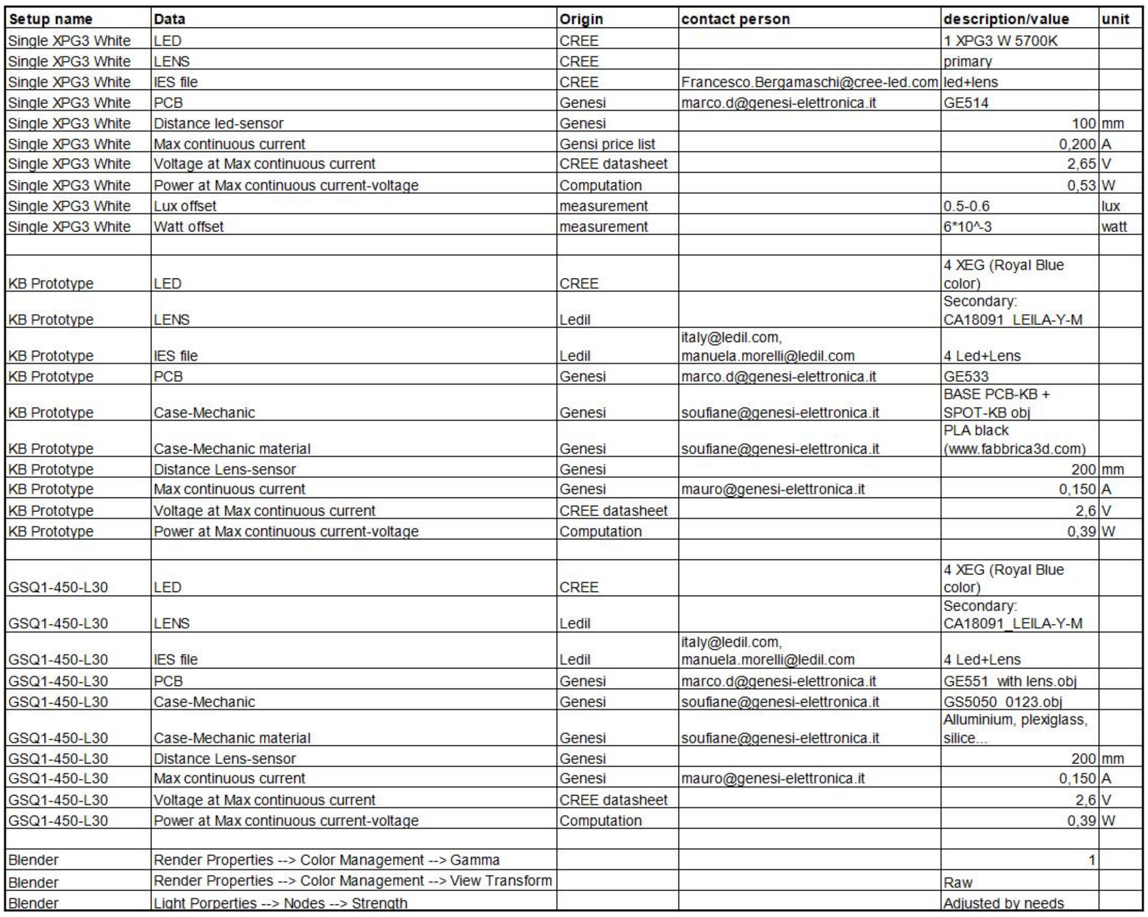


Figure 7. Experimental Setup Overview



Setup name	Data	Origin	contact person	description/value	unit
Single XPG3 White	LED	CREE		1 XPG3 W 5700K	
Single XPG3 White	LENS	CREE		primary	
Single XPG3 White	IES file	CREE	Francesco.Bergamaschi@cree-led.com	led+lens	
Single XPG3 White	PCB	Genesi	marco.d@genesi-elettronica.it	GE514	
Single XPG3 White	Distance led-sensor	Genesi		100	mm
Single XPG3 White	Max continuous current	Gensi price list		0,200	A
Single XPG3 White	Voltage at Max continuous current	CREE datasheet		2,65	V
Single XPG3 White	Power at Max continuous current-voltage	Computation		0,53	W
Single XPG3 White	Lux offset	measurement		0.5-0.6	lux
Single XPG3 White	Watt offset	measurement		6*10^-3	watt
KB Prototype	LED	CREE		4 XEG (Royal Blue color)	
KB Prototype	LENS	Ledil		Secondary: CA18091_LEILA-Y-M	
KB Prototype	IES file	Ledil	italy@ledil.com, manuela.morelli@ledil.com	4 Led+Lens	
KB Prototype	PCB	Genesi	marco.d@genesi-elettronica.it	GE533	
KB Prototype	Case-Mechanic	Genesi	soufiane@genesi-elettronica.it	BASE PCB-KB + SPOT-KB obj	
KB Prototype	Case-Mechanic material	Genesi	soufiane@genesi-elettronica.it	PLA black (www.fabbrica3d.com)	
KB Prototype	Distance Lens-sensor	Genesi		200	mm
KB Prototype	Max continuous current	Genesi	mauro@genesi-elettronica.it	0,150	A
KB Prototype	Voltage at Max continuous current	CREE datasheet		2,6	V
KB Prototype	Power at Max continuous current-voltage	Computation		0,39	W
GSQ1-450-L30	LED	CREE		4 XEG (Royal Blue color)	
GSQ1-450-L30	LENS	Ledil		Secondary: CA18091_LEILA-Y-M	
GSQ1-450-L30	IES file	Ledil	italy@ledil.com, manuela.morelli@ledil.com	4 Led+Lens	
GSQ1-450-L30	PCB	Genesi	marco.d@genesi-elettronica.it	GE551_with lens.obj	
GSQ1-450-L30	Case-Mechanic	Genesi	soufiane@genesi-elettronica.it	GS5050_0123.obj	
GSQ1-450-L30	Case-Mechanic material	Genesi	soufiane@genesi-elettronica.it	Alluminium, plexiglass, silice...	
GSQ1-450-L30	Distance Lens-sensor	Genesi		200	mm
GSQ1-450-L30	Max continuous current	Genesi	mauro@genesi-elettronica.it	0,150	A
GSQ1-450-L30	Voltage at Max continuous current	CREE datasheet		2,6	V
GSQ1-450-L30	Power at Max continuous current-voltage	Computation		0,39	W
Blender	Render Properties --> Color Management --> Gamma			1	
Blender	Render Properties --> Color Management --> View Transform			Raw	
Blender	Light Porperties --> Nodes --> Strength			Adjusted by needs	

Figure 8. Parameter Settings Used During Testing

In general, during my internship in the R&D department of Genesi Elettronica, I undertook tasks ranging from theoretical research, system design, software development, experimental testing, data collection and analysis. By participating in these tasks, I not only mastered the core theoretical and practical skills of machine vision system virtualization and simulation technology, but also accumulated rich engineering experience in project practice, and laid a solid foundation for further exploration in the field of intelligent manufacturing and industrial automation in the future.

5.2 Challenges and Solutions

In the actual operation process, I encountered many challenges, which have influenced the rendering quality, data accuracy and calculation efficiency. Through analysis and experimental verification of these problems, I gradually found solutions and continuously optimized the simulation system to improve its reliability and practicality in industrial lighting simulation.

First of all, the light intensity distribution of virtual light failed to match the actual light, which has become the primary problem of lighting simulation. In the initial experiments, when using the Cycles engine, although the IES file can provide the photometric data of the light source, the rendered lighting distribution was still significantly different from the real light source. Specifically, the attenuation speed of light was different, the lighting range in the virtual environment was relatively small, the edge of the light spot is too sharp, and the actual measured lighting area is more uniform. This difference was mainly due to the simplified processing of the rendering engine to calculate the propagation of light. In order to solve this problem, it was necessary to adjust the basic parameters in the rendering settings interface in Blender, and select the appropriate parameters to make the simulated light close to the actual light effect. The specific operation was to select 'Raw' in the 'View Transform' option in 'Color Management', and adjust 'Gamma' to 1. After several rounds of experimental adjustments, I successfully reduced the error between the virtual light source and the actual light source, making the light intensity distribution of the two tend to be consistent, therefore improving the accuracy of the overall lighting simulation.

Secondly, the choice of background material played a key role in lighting simulation. During the experiment, I found that background plates of different materials will significantly affect the reflection and scattering of light, therefore changing the distribution of light. For example, the lower the roughness of the background surface in the virtual scene, the weaker the attenuation of light and light intensity with distance, making the overall brightness of the area close to the center point higher. Therefore, it was necessary to adjust the material parameters in Blender by setting metallic to 0 and roughness to 1, so that it could match the optical properties of the real background plate and accurately reflect the light distribution in the real scene without causing additional interference to the overall light propagation.

In addition to the distribution of light and the selection of materials, during the data collection process, when the light source power was too high, the pixel value was prone to overflow. Due to the limited range of pixel values, in the high light intensity area, the brightness value of some pixels reached the maximum value, resulting in the loss of detail information in the highlight part of the image. This phenomenon made it difficult to conduct effective data analysis in the area close to the center of the light source, because all pixel values tended to be saturated and could not reflect the actual changes in light intensity. To solve this problem, after importing the IES file, could use the node where the IES file was located to adjust the intensity, so that the overflow problem of pixel values could be solved without changing the basic characteristics and distribution of the light (Figure 9).

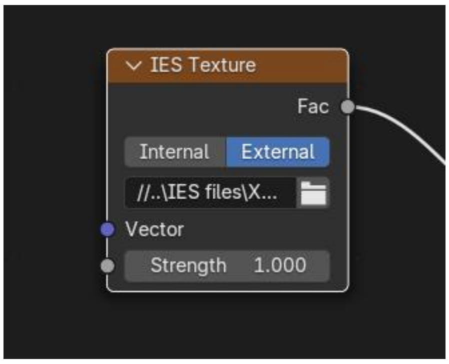


Figure 9. Adjusting IES Light Intensity via Node Editor to Prevent Pixel Overflow

Optimizing rendering time was also an important challenge. When LuxCoreRender was selected as the rendering engine, due to its high-quality rendering effect, the rendering time of each image was too long, which seriously affected the efficiency of the experiment. In some test scenarios, the calculation time of a high-resolution rendered image even exceeded 30 minutes, which was unacceptable for experiments that require a lot of data collection and analysis. In order to improve rendering efficiency, the maximum rendering time was set, so that after reaching the time limit, the rendering would automatically stop and the image would be exported. After multiple rendering image comparisons, the maximum rendering time was set to 300 seconds, which balanced the rendering quality and calculation time. While greatly reducing the rendering time, the image details remained within an acceptable range.

Finally, Blender's default IES light source could not change the color of light, which imposed certain limitations on simulating different types of light sources. In subsequent applications, it might be necessary to test light sources with different wavelength combinations, but IES files could only control the direction and intensity of light, but couldn't directly define the color of light. To solve this problem, LuxCoreRender could be used for spectral rendering. This renderer allowed users to customize the spectral distribution of light sources, therefore achieving more precise color temperature control. The specific operation was to add the 'Irregular Data' node to the Nodes setting of the light source control panel (Figure 10), which could simulate lights with different wavelength combinations, but this method also had limitations, which was only applicable to the LuxCoreRender engine. For the Cycles engine, there was currently no particularly good way to accurately simulate light sources of different colors.

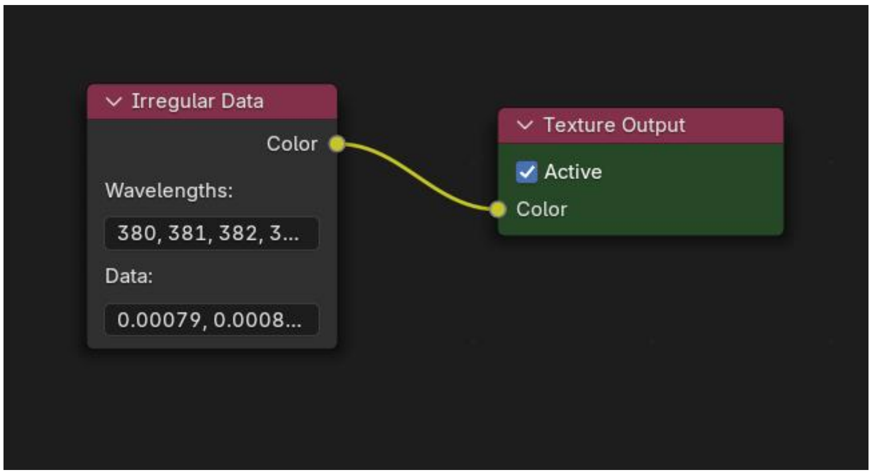

Irregular Data
Color
Wavelengths:
380, 381, 382, 3...
Data:
0.00079, 0.0008...
Texture Output
Active
Color

Figure 10. Customizing Spectral Distribution Using the 'Irregular Data' Node in LuxCoreRender

In summary, during this internship, I encountered several technical challenges, including inconsistent lighting distribution, background material influence, rendering time optimization, pixel value overflow, and IES light source color adjustment. In response to these challenges, I gradually found effective solutions through analysis and experimental optimization, and improved the accuracy and practicality of the virtual simulation platform through continuous debugging and improvement. These research results not only improved the accuracy of lighting simulation, but also laid a solid foundation for subsequent optimization and research.

5.3 Further Virtual Lighting System Construction

With the development of Machine Vision technology, virtual lighting systems have become an important tool for creating high-quality commercial visual content. Based on the powerful functions of the Blender rendering engine, this study is committed to building a highly realistic virtual lighting system to meet the needs of magazine cover design. The image rendered this time shows a lighting system with a sense of technology. Its key design concept is to create immersive light and shadow effects while enhancing visual impact. It is hoped that a new visual experience can be provided.

In this rendering project, the core light source in the image was composed of multiple linear light sources, and a global illumination algorithm based on path tracing was used to ensure the natural propagation of light and reduce unnecessary hard-edge shadows. In addition, this project focused on the construction and optimization of the lighting system, and adopted a physically based lighting simulation method to ensure the authenticity and fineness of the lighting effects. In addition, the angle and illumination direction of the lighting system were finely adjusted to ensure that the light beam could evenly cover the background area, so that the text content on the background board could be clearly illuminated-"Lighting for Machine Vision". This lighting design could effectively reduce the strong shadows in the picture, making the overall effect more balanced and natural (Figure 11).

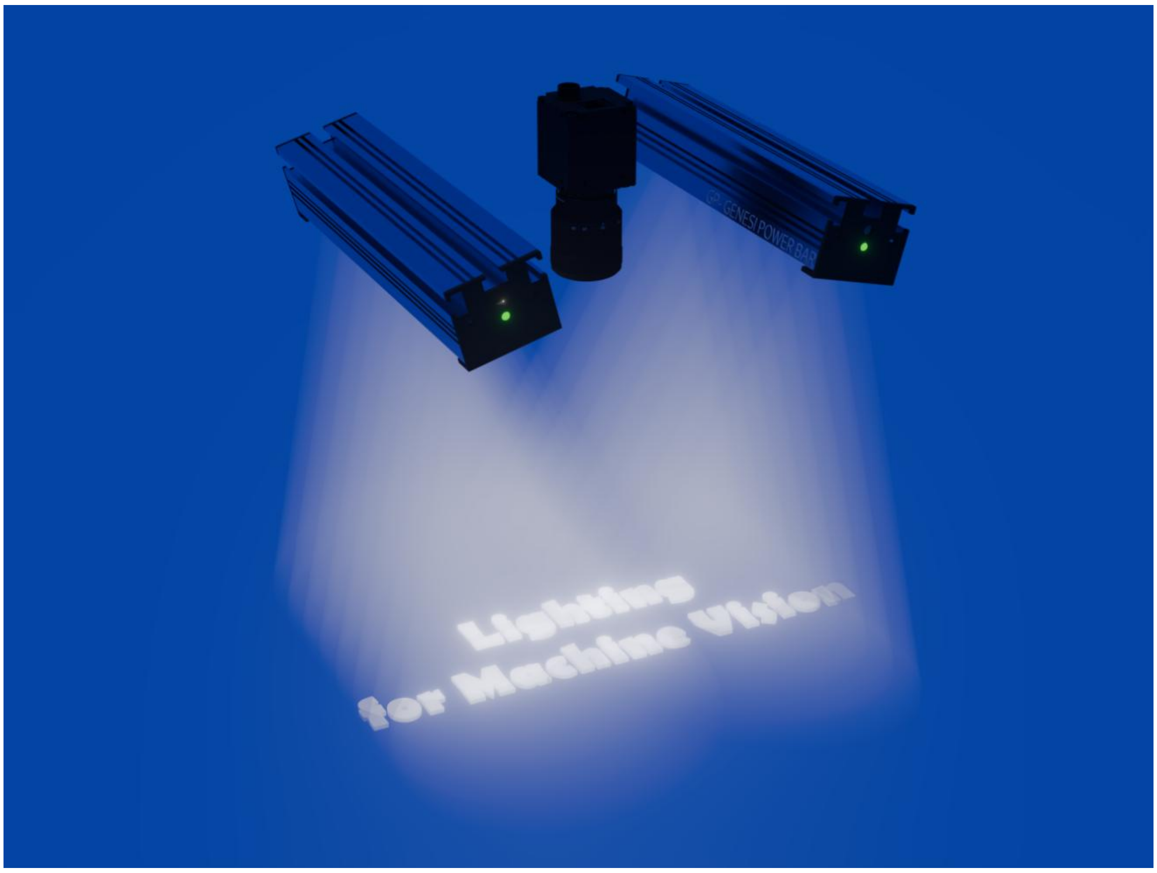


Figure 11. Optimized Lighting Design for a Softer and More Natural Appearance

In traditional commercial photography, the adjustment of the lighting system often requires a large amount of physical equipment and a complex lighting process, while the introduction of virtual lighting technology has greatly improved work efficiency and reduced production costs. In order to ensure the final image quality, the adaptive sampling technology (Adaptive Sampling) was used, that allowed rendering calculations to be concentrated in key areas rather than evenly distributing computing resources. This optimization strategy significantly improved rendering efficiency while reducing rendering noise, making the final image clearer and more delicate.

6. Results and Analysis

6.1 Data Presentation

During this internship, research on the construction and optimization of the virtual lighting system for machine vision was conducted, and a large amount of data related to lighting simulation through multiple experiments was collected. In this study, the data presented were obtained through a structured and repeatable process that combines virtual simulation and real experimental measurements. For virtual data acquisition, two rendering engines (Cycles and LuxCoreRender) in Blender were used, and controlled with a custom-written Python automation script. The script first sets the rendering parameters and finally extracts the pixel values of key areas in the final rendered image, unifying the resolution and color depth settings. The final exported rendering image resolution is 800×600 pixels, and the pixel values are read in a 400×300 coordinate system based on a horizontal and vertical interval of 30mm, ensuring sampling accuracy and consistency.

For the acquisition of physical data, a real experimental platform corresponding to the virtual scene needs to be built in the laboratory, and the key to this is the precise placement of the background plate and the light source device. Two sensor tools were used in the experiment for data acquisition: an **illuminance sensor** (used to measure illuminance in Lux) and a **light intensity sensor** (used to measure the radiation intensity in a specific direction in $\mathrm{Watt/m^2}$). The two sensors were placed at the key reference points of the horizontal and vertical coordinates of the background plate, and multiple measurements were performed on each measurement point to reduce random errors and improve data reliability. During the measurement process, the natural light and stray light in the experimental environment were strictly controlled to ensure that the measurement results were not interfered by external light sources.

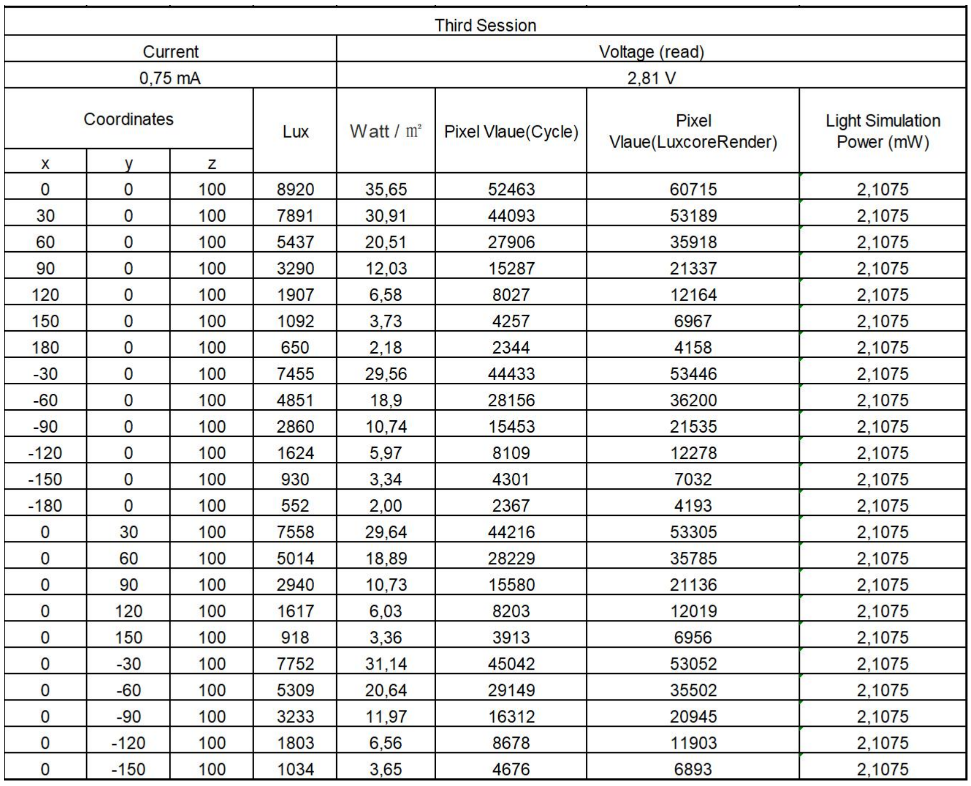
Through the combination of automated pixel extraction and physical sensor measurement, this study established a stable and reliable data acquisition system. The following is the data presentation results obtained during the internship.

First Session							
Current				Voltage (read)			
0,2 mA				2,65 V			
Coordinates			Lux	Watt / ㎡	Pixel Vlaue(Cycle)	Pixel Vlaue(LuxcoreRender)	Light Simulation Power (mW)
x(mm)	y(mm)	z(mm)					
0	0	100	2692	10,52	13194	15266	0,53
30	0	100	2353	9,07	11089	13376	0,53
60	0	100	1620	6,00	7018	9032	0,53
90	0	100	980	3,48	3845	5366	0,53
120	0	100	557	1,91	2019	3059	0,53
150	0	100	322	1,08	1071	1752	0,53
180	0	100	188	0,64	589	1045	0,53
-30	0	100	2267	8,79	11149	13440	0,53
-60	0	100	1502	5,66	7081	9103	0,53
-90	0	100	869	3,21	3886	5416	0,53
-120	0	100	496	1,79	2039	3088	0,53
-150	0	100	281	1,00	1082	1768	0,53
-180	0	100	168	0,59	595	1054	0,53
0	30	100	2280	8,73	11119	13406	0,53
0	60	100	1492	5,56	7099	9000	0,53
0	90	100	868	3,14	3918	5316	0,53
0	120	100	484	1,73	2063	3022	0,53
0	150	100	273	0,97	984	1749	0,53
0	-30	100	2347	9,18	11327	13341	0,53
0	-60	100	1582	6,11	7330	8929	0,53
0	-90	100	955	3,53	4102	5267	0,53
0	-120	100	533	1,94	2182	2993	0,53
0	-150	100	306	1,08	1176	1733	0,53

Table 1. Comparison of Simulated and Measured Lighting Data at Multiple Coordinates (0.2 mA, 2.65 V)

Second Session							
Current				Voltage (read)			
0,5 mA				2,74 V			
Coordinates			Lux	Watt / ㎡	Pixel Vlaue(Cycle)	Pixel Vlaue(LuxcoreRender)	Light Simulation Power (mW)
x	y	z					
0	0	100	6274	24,93	34104	39449	1,37
30	0	100	5512	21,45	28663	34575	1,37
60	0	100	3781	14,2	18141	23346	1,37
90	0	100	2288	8,22	9938	13871	1,37
120	0	100	1310	4,53	5218	7910	1,37
150	0	100	754	2,53	2767	4529	1,37
180	0	100	444	1,49	1524	2702	1,37
-30	0	100	5268	20,82	28819	34740	1,37
-60	0	100	3487	13,34	18303	2530	1,37
-90	0	100	2025	7,50	10045	14000	1,37
-120	0	100	1137	4,19	5271	790	1,37
-150	0	100	646	2,32	2796	4570	1,37
-180	0	100	385	1,37	1538	2725	1,37
0	30	100	5281	20,66	28743	34648	1,37
0	60	100	3493	13,31	18351	23263	1,37
0	90	100	2023	7,48	10128	13737	1,37
0	120	100	1123	4,09	5332	7812	1,37
0	150	100	631	2,28	2544	4522	1,37
0	-30	100	5490	21,78	29280	34488	1,37
0	-60	100	3744	14,42	18949	23079	1,37
0	-90	100	2249	8,35	10604	1617	1,37
0	-120	100	1257	4,51	5641	7737	1,37
0	-150	100	718	2,52	3039	4481	1,37

Table 2. Comparison of Simulated and Measured Lighting Data at Multiple Coordinates (0.5 mA, 2.74 V)



Third Session							
Current				Voltage (read)			
0,75 mA				2,81 V			
Coordinates			Lux	Watt / ㎡	Pixel Vlaue(Cycle)	Pixel Vlaue(LuxcoreRender)	Light Simulation Power (mW)
x	y	z					
0	0	100	8920	35,65	52463	60715	2,1075
30	0	100	7891	30,91	44093	53189	2,1075
60	0	100	5437	20,51	27906	35918	2,1075
90	0	100	3290	12,03	15287	21337	2,1075
120	0	100	1907	6,58	8027	12164	2,1075
150	0	100	1092	3,73	4257	6967	2,1075
180	0	100	650	2,18	2344	4158	2,1075
-30	0	100	7455	29,56	44433	53446	2,1075
-60	0	100	4851	18,9	28156	36200	2,1075
-90	0	100	2860	10,74	15453	21535	2,1075
-120	0	100	1624	5,97	8109	12278	2,1075
-150	0	100	930	3,34	4301	7032	2,1075
-180	0	100	552	2,00	2367	4193	2,1075
0	30	100	7558	29,64	44216	53305	2,1075
0	60	100	5014	18,89	28229	35785	2,1075
0	90	100	2940	10,73	15580	21136	2,1075
0	120	100	1617	6,03	8203	12019	2,1075
0	150	100	918	3,36	3913	6956	2,1075
0	-30	100	7752	31,14	45042	53052	2,1075
0	-60	100	5309	20,64	29149	35502	2,1075
0	-90	100	3233	11,97	16312	20945	2,1075
0	-120	100	1803	6,56	8678	11903	2,1075
0	-150	100	1034	3,65	4676	6893	2,1075

Table 3. Comparison of Simulated and Measured Lighting Data at Multiple Coordinates (0.75 mA, 2.81 V)

In the process of lighting simulation and actual physical model measurement, the relationship between real illuminance, light intensity and pixel value was studied, so that the pixel value could be used to predict illuminance and light intensity in subsequent analysis. In the experiment, multiple different measurement points were selected in the rendering scene, and the corresponding relationship between their pixel values and illuminance and light intensity was recorded.

6.2 Analysis and Interpretation

In the data analysis, we conducted an in-depth analysis of the relationship between the pixel values read during the lighting simulation and the illumination and intensity of the actual physical model, and explored the impact of the IES file on the lighting effect. Through the collection and modeling of experimental data, we hope to be able to predict the actual lighting effect based on the virtual lighting model and optimize the accuracy of the lighting simulation.

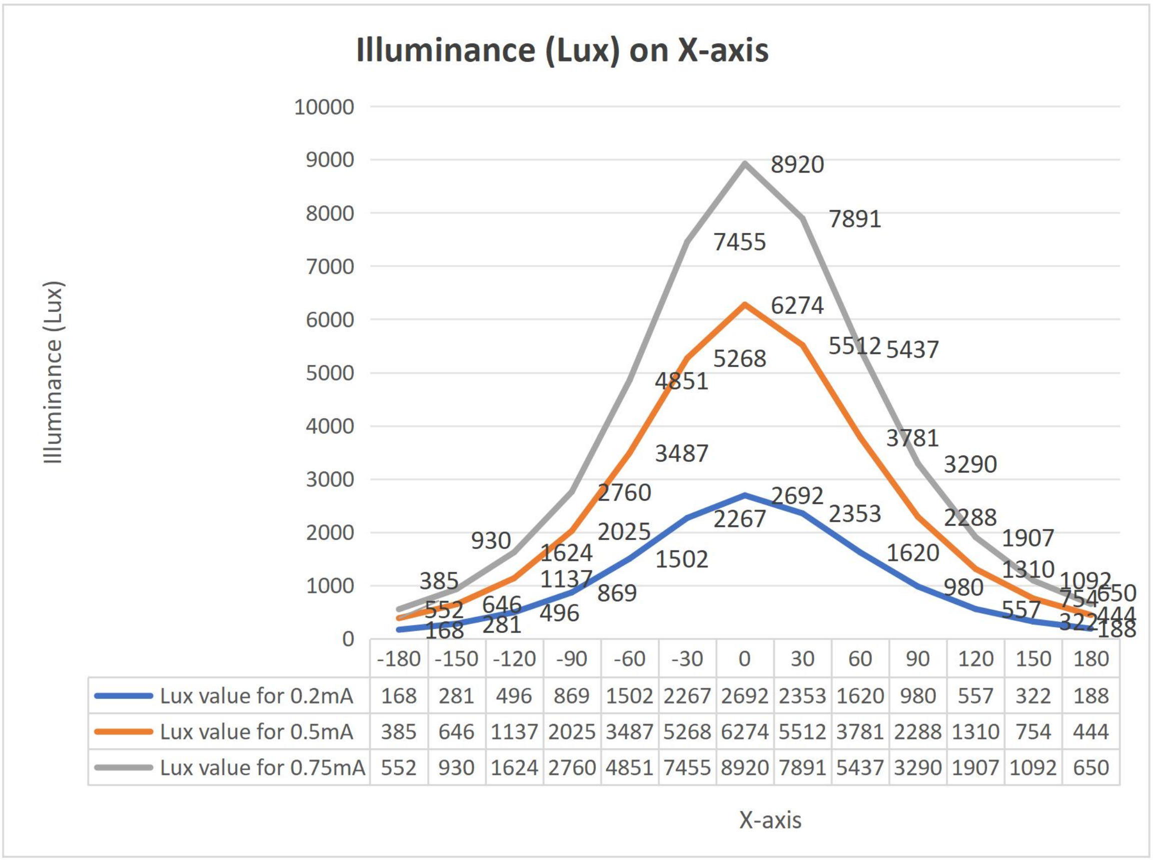
6.2.1 Analysis for Real Light Behaviour

In order to support the analysis of real lighting behavior, all the measured data collected in laboratory environment and the experimental environment layout were consistent with the virtual simulation platform. The size of the background board used in the experiment was 400 mm × 300 mm, which corresponds to the coordinate system in the virtual model. The point where the light source is vertically projected onto the background board is taken as the origin (0,0), and data sampling is performed every 30 mm along the x-axis and y-axis.

During the data collection process, two calibrated sensors were used: the illuminance sensor is used to record the illuminance value in Lux in the unit direction, and the light intensity sensor is used to measure the radiant flux in Watt/m^2 in the unit direction. These two sensors are fixed at the specified coordinate points on the surface of the background board to ensure the repeatability and consistency of each acquisition. In order to minimize the interference of ambient light, the experiment was carried out under low ambient light conditions, and multiple readings were taken at each sampling point, and the average value was finally taken to improve the accuracy of the data and reduce the impact of accidental errors.

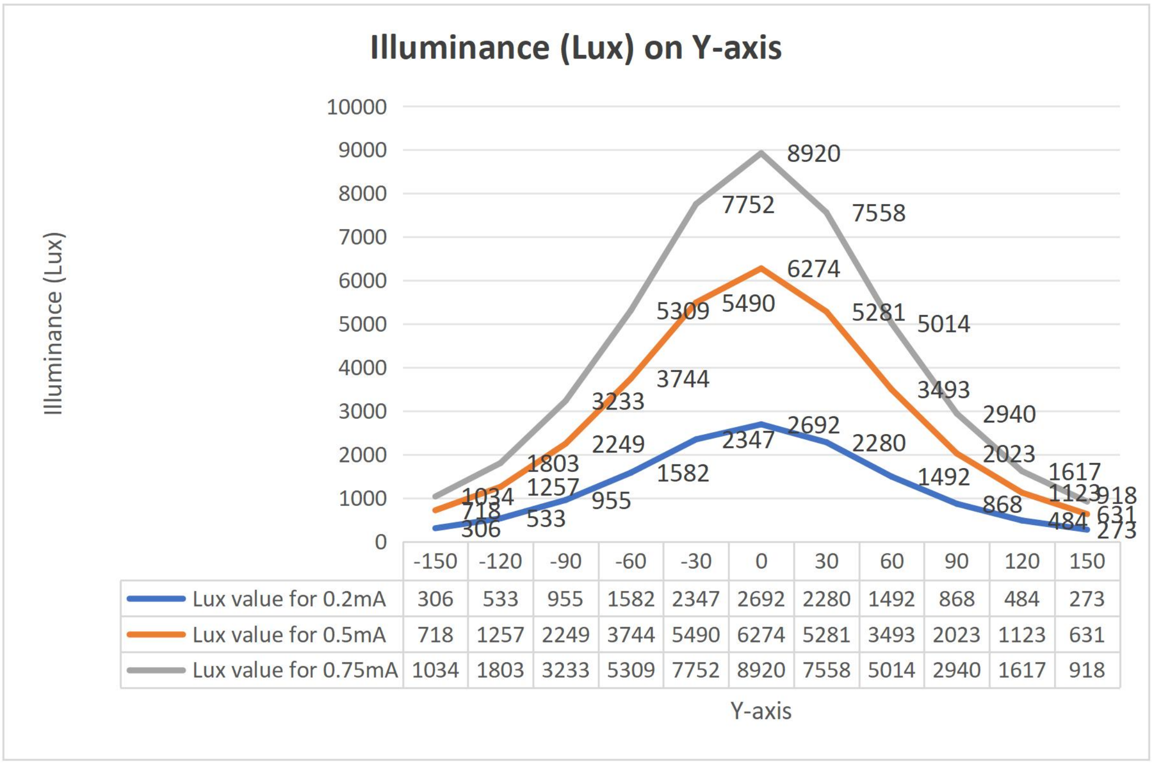
The collected raw data is structured and processed to form a standard data table containing plane coordinates (x, y) and illuminance value (Lux), plane coordinates (x, y) and radiation intensity (Watt/m^2). This data set is then compared and analyzed with the pixel values of the corresponding positions in the virtual rendering output image, in order to effectively evaluate the real restoration ability and physical accuracy of the simulated lighting system.

Here are the charts that show the relationship between illuminance and distance.



X-axis	-180	-150	-120	-90	-60	-30	0	30	60	90	120	150	180
Lux value for 0.2mA	168	281	496	869	1502	2267	2692	2353	1620	980	557	322	188
Lux value for 0.5mA	385	646	1137	2025	3487	5268	6274	5512	3781	2288	1310	754	444
Lux value for 0.75mA	552	930	1624	2760	4851	7455	8920	7891	5437	3290	1907	1092	650

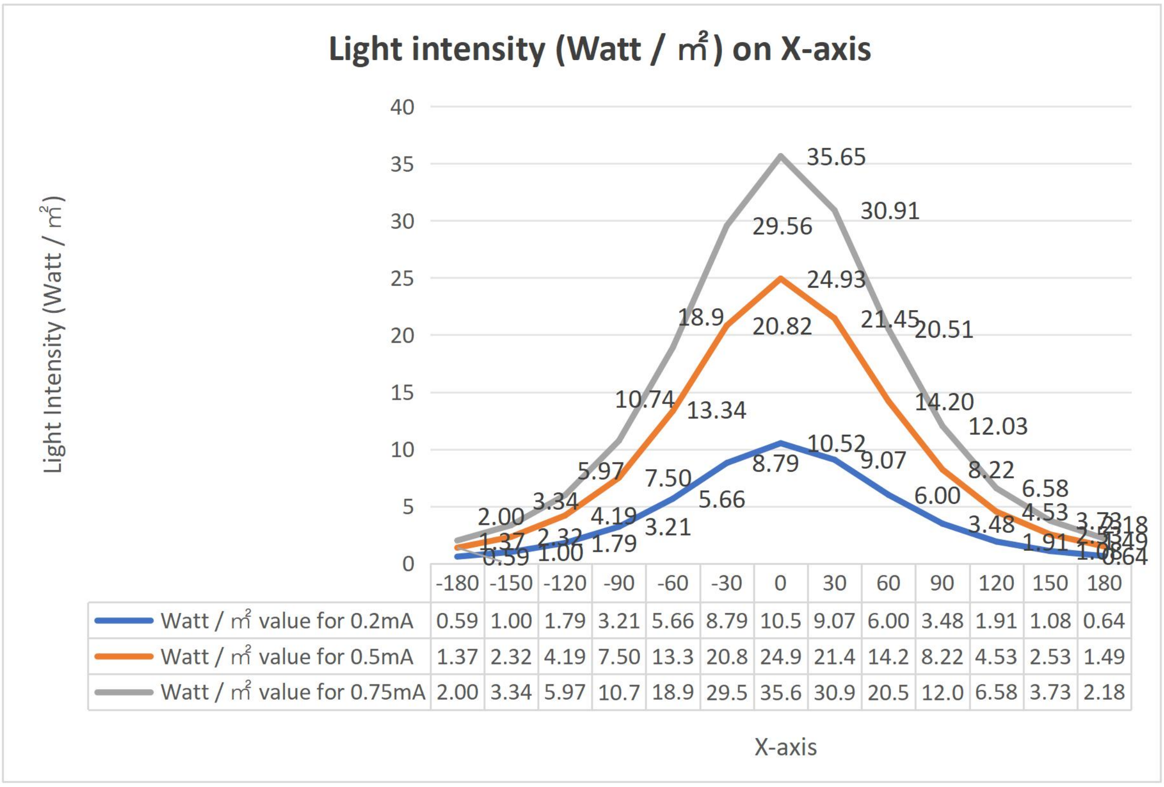
Figure 12. Illuminance Distribution Along the X-Axis at 0.2 mA, 0.5 mA, and 0.75 mA



Y-axis	-150	-120	-90	-60	-30	0	30	60	90	120	150
Lux value for 0.2mA	306	533	955	1582	2347	2692	2280	1492	868	484	273
Lux value for 0.5mA	718	1257	2249	3744	5490	6274	5281	3493	2023	1123	631
Lux value for 0.75mA	1034	1803	3233	5309	7752	8920	7558	5014	2940	1617	918

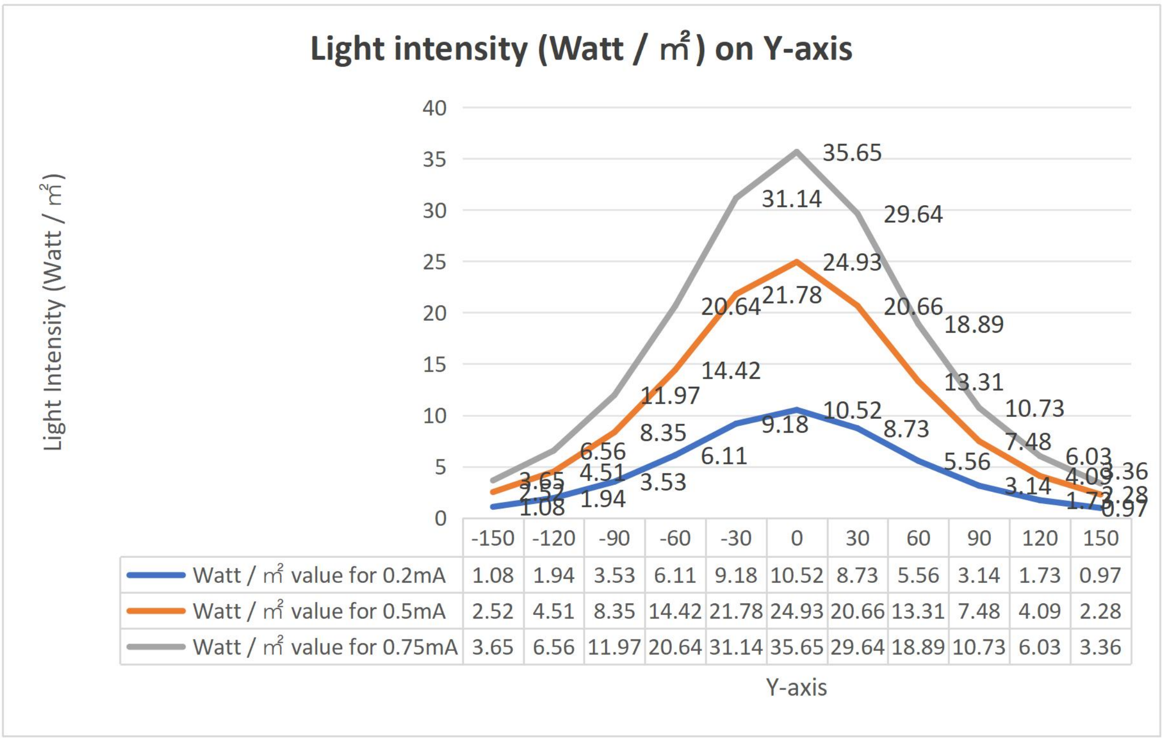
Figure 13. Illuminance Distribution Along the Y-Axis at 0.2 mA, 0.5 mA, and 0.75 mA

Here are the charts that show the relationship between light intensity and distance.



	-180	-150	-120	-90	-60	-30	0	30	60	90	120	150	180
Watt / m² value for 0.2mA	0.59	1.00	1.79	3.21	5.66	8.79	10.5	9.07	6.00	3.48	1.91	1.08	0.64
Watt / m² value for 0.5mA	1.37	2.32	4.19	7.50	13.3	20.8	24.9	21.4	14.2	8.22	4.53	2.53	1.49
Watt / m² value for 0.75mA	2.00	3.34	5.97	10.7	18.9	29.5	35.6	30.9	20.5	12.0	6.58	3.73	2.18

Figure 14. Light Intensity Distribution Along the X-Axis at 0.2 mA, 0.5 mA, and 0.75 mA



	-150	-120	-90	-60	-30	0	30	60	90	120	150
Watt / m² value for 0.2mA	1.08	1.94	3.53	6.11	9.18	10.52	8.73	5.56	3.14	1.73	0.97
Watt / m² value for 0.5mA	2.52	4.51	8.35	14.42	21.78	24.93	20.66	13.31	7.48	4.09	2.28
Watt / m² value for 0.75mA	3.65	6.56	11.97	20.64	31.14	35.65	29.64	18.89	10.73	6.03	3.36

Figure 15. Light Intensity Distribution Along the Y-Axis at 0.2 mA, 0.5 mA, and 0.75 mA

From these four charts, it can be seen that the distribution of light illuminance and light intensity presents a symmetrical bell-shaped curve, that is, the light intensity is highest in the center area and gradually decreases to both sides. From the shape of the curve, they are similar to Gaussian distribution.

6.2.2 Analysis for Cycles Engine

The first ones shown are the rendered images when Cycles is selected as the rendering engine (Figure 16, 17 and 18). The light source powers in the simulated environment are 0.53 mW, 1.37 mW and 2.1075 mW respectively.

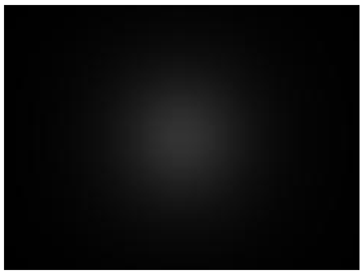


Figure 16. Rendering Light Distribution at 0.53 mW (Cycles)

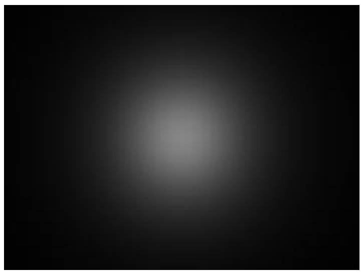


Figure 17. Rendering Light Distribution at 1.37 mW (Cycles)

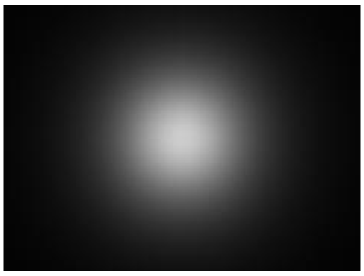
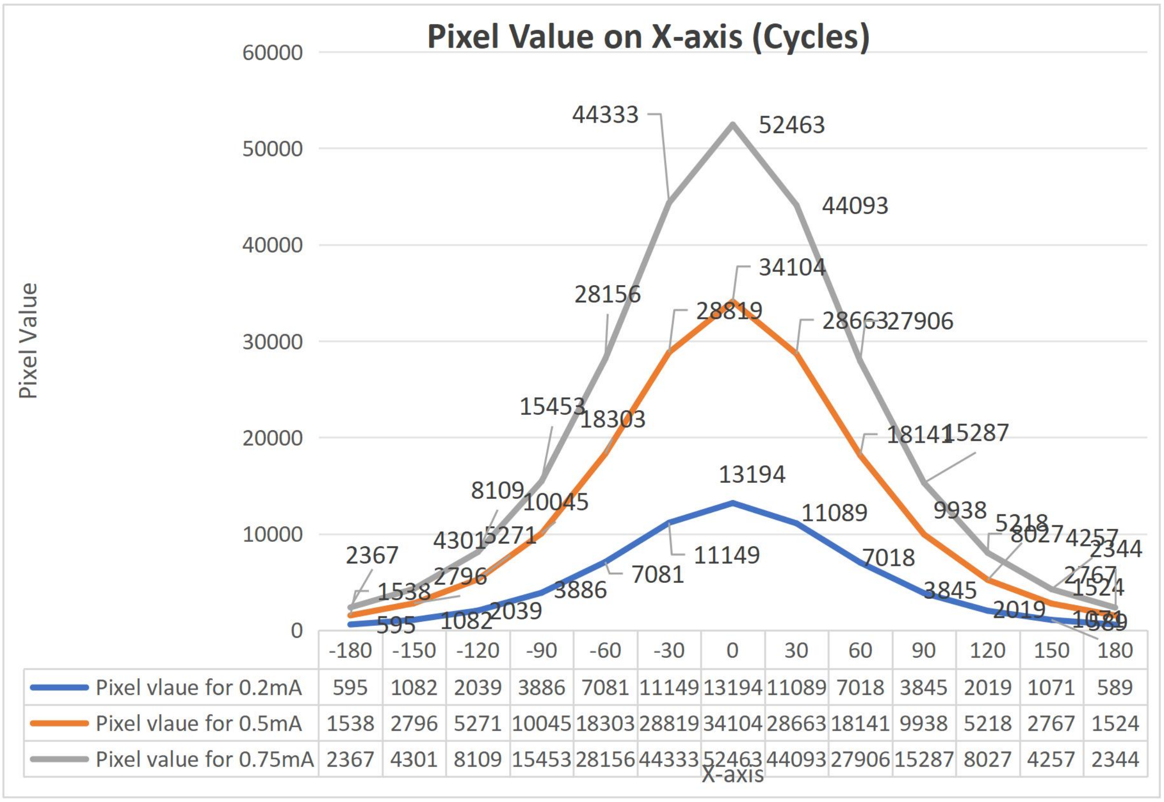


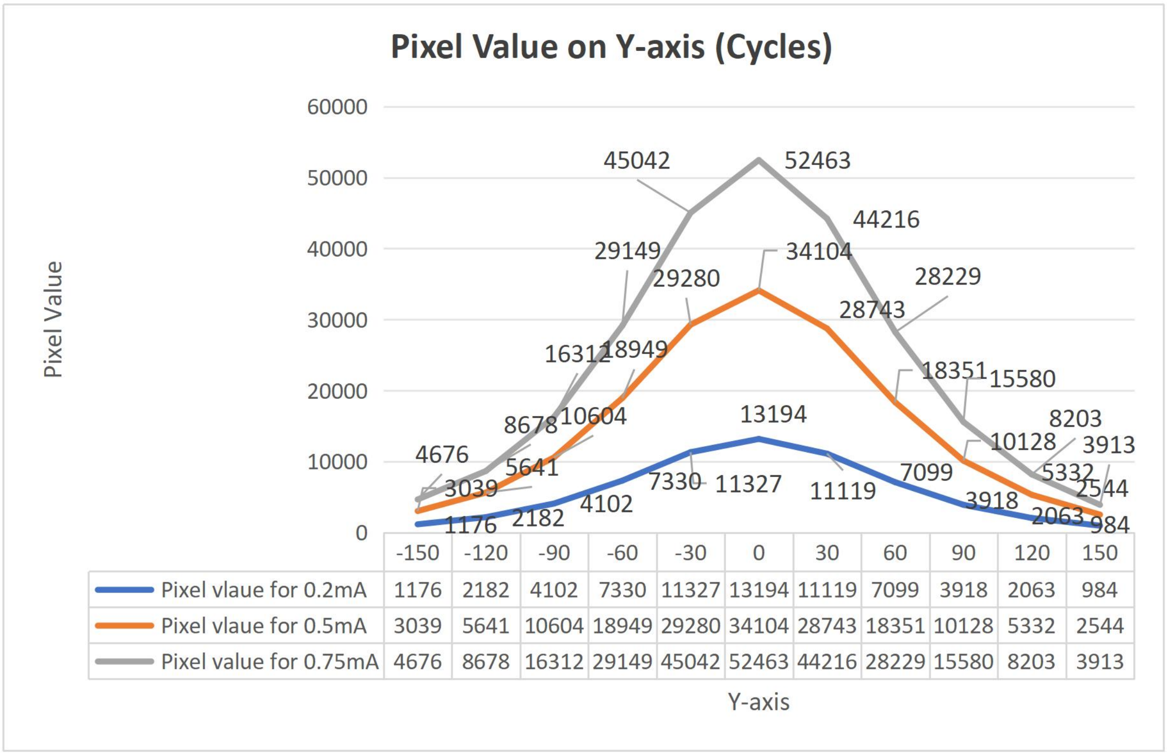
Figure 18. Rendering Light Distribution at 2.1075 mW (Cycles)

The following charts show the relationship between image pixel value and distance both on X-axis, Y-axis when using the Cycles engine.



	-180	-150	-120	-90	-60	-30	0	30	60	90	120	150	180
Pixel vlaue for 0.2mA	595	1082	2039	3886	7081	11149	13194	11089	7018	3845	2019	1071	589
Pixel vlaue for 0.5mA	1538	2796	5271	10045	18303	28819	34104	28663	18141	9938	5218	2767	1524
Pixel value for 0.75mA	2367	4301	8109	15453	28156	44333	52463	44093	27906	15287	8027	4257	2344

Figure 19. Pixel Value Distribution Along the X-Axis at 0.2 mA, 0.5 mA, and 0.75 mA (Cycles)



	-150	-120	-90	-60	-30	0	30	60	90	120	150
Pixel vlaue for 0.2mA	1176	2182	4102	7330	11327	13194	11119	7099	3918	2063	984
Pixel vlaue for 0.5mA	3039	5641	10604	18949	29280	34104	28743	18351	10128	5332	2544
Pixel value for 0.75mA	4676	8678	16312	29149	45042	52463	44216	28229	15580	8203	3913

Figure 20. Pixel Value Distribution Along the Y-Axis at 0.2 mA, 0.5 mA, and 0.75 mA (Cycles)

From these two charts, the vertical axis represents the pixel value and the horizontal axis represents the distance. The pixel value shows a symmetrical peak distribution with the change of distance, and the peak appears in the central area (0 position). Different currents (0.2mA, 0.5mA, 0.75mA) correspond to different curves. The larger the current, the higher the peak value, which means that higher current drive will make the light brighter, resulting in an increase in pixel value. The current value here corresponds to the current value used in the three tests in the actual physical model. It can also be observed that at positions far away from the center (±150 and ±180), the pixel value decays rapidly, and also shows the characteristics of a Gaussian distribution, which is similar to the relationship between illuminance, light intensity and distance in Figure 12, 13, 14 and 15.

Now, based on the actual illuminance curve, actual light intensity curve, and the pixel value curve obtained by the Cycles engine, since their distribution forms are similar, it can be inferred that there is a linear relationship between them. The following four charts show the possible linear relationship between them (Figure 21, 22, 23 and 24).

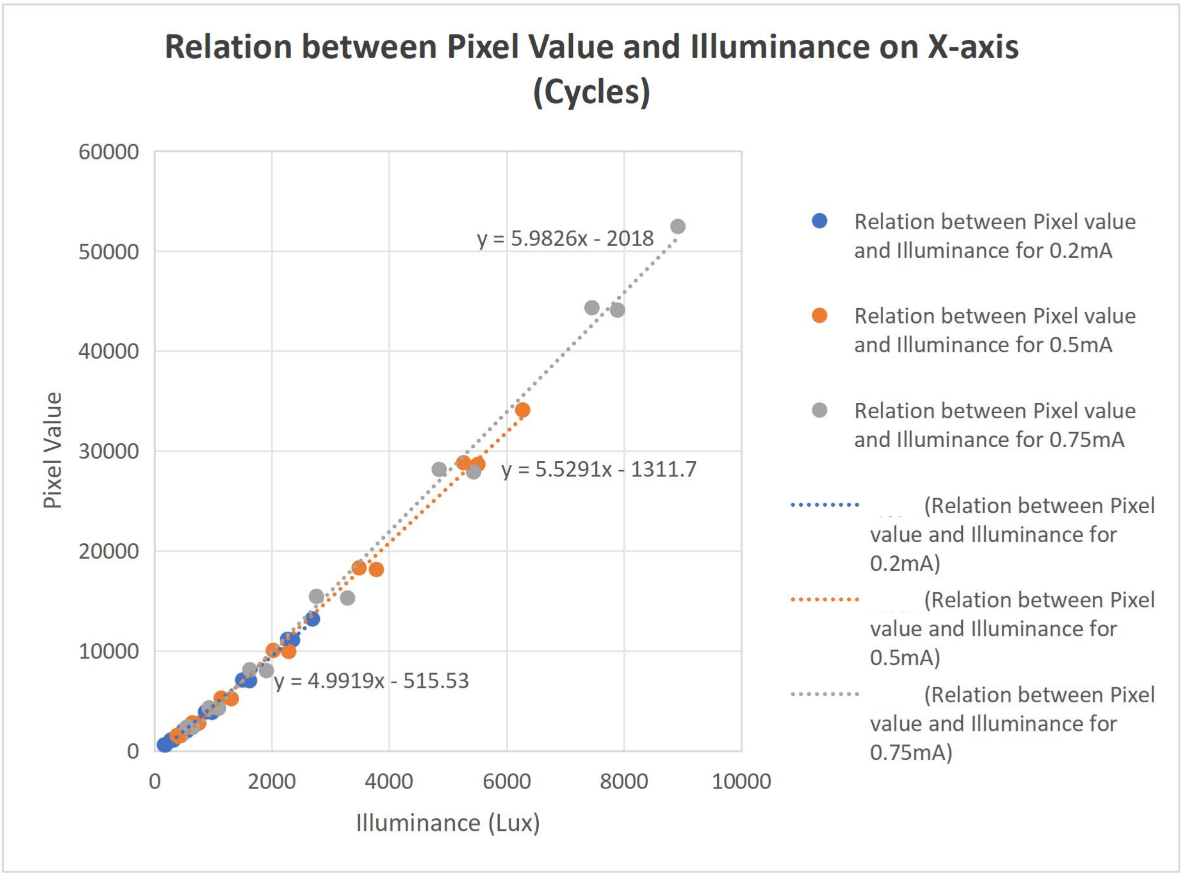


Figure 21. Relation between Pixel Value and Illuminance on X-axis

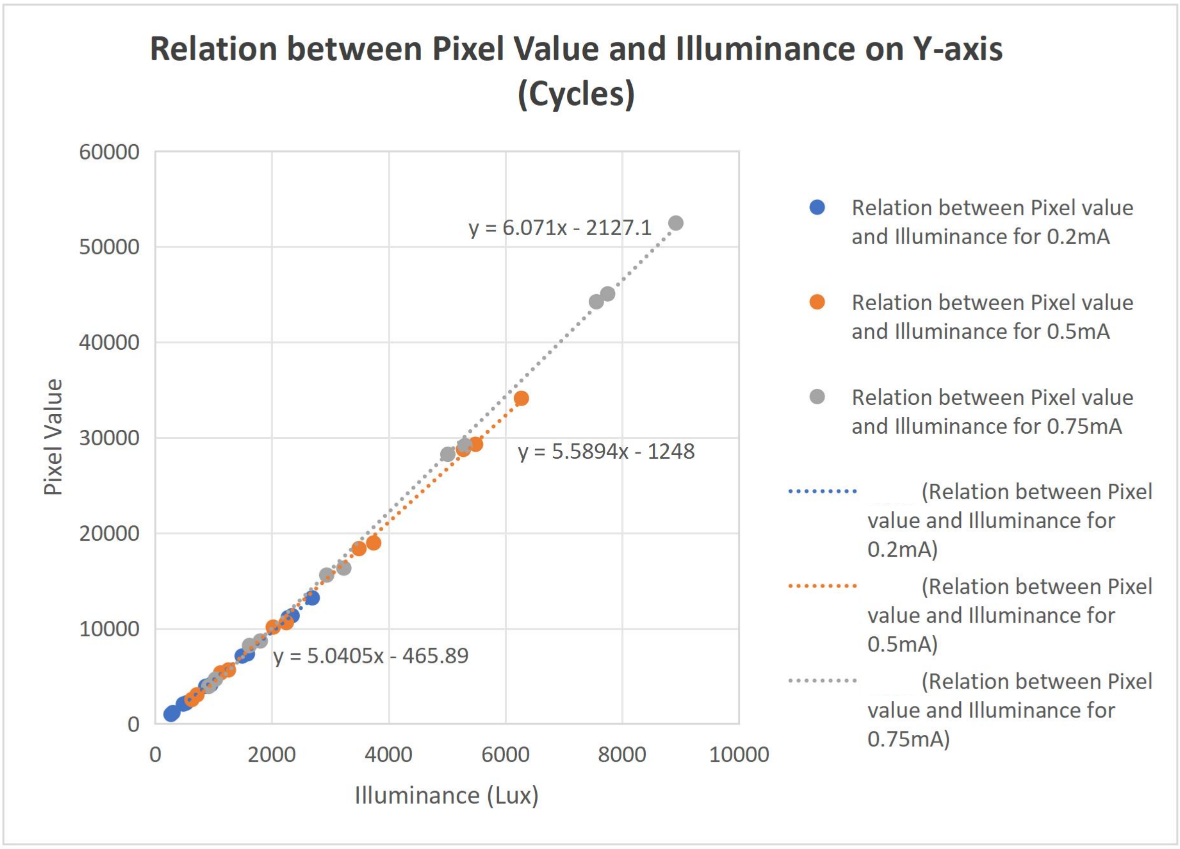


Figure 22. Relation between Pixel Value and Illuminance on Y-axis

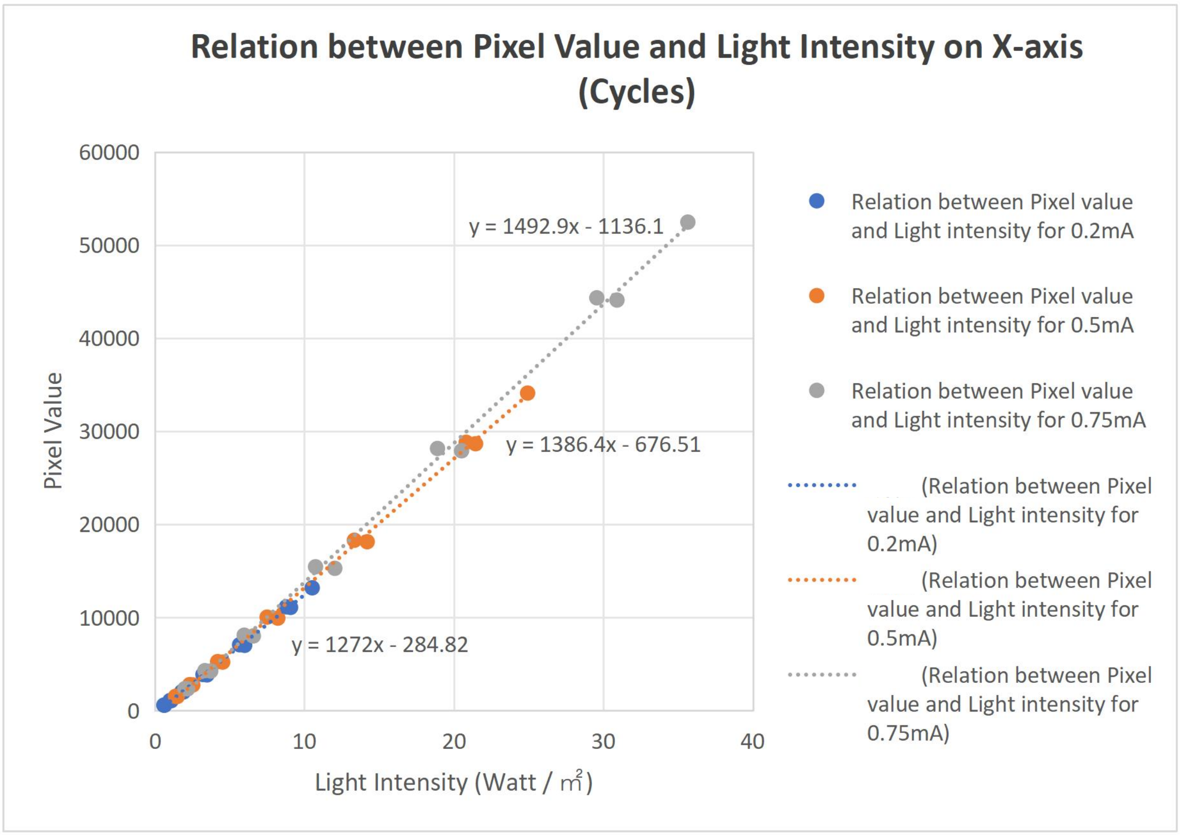


Figure 23. Relation between Pixel Value and Light Intensity on X-axis

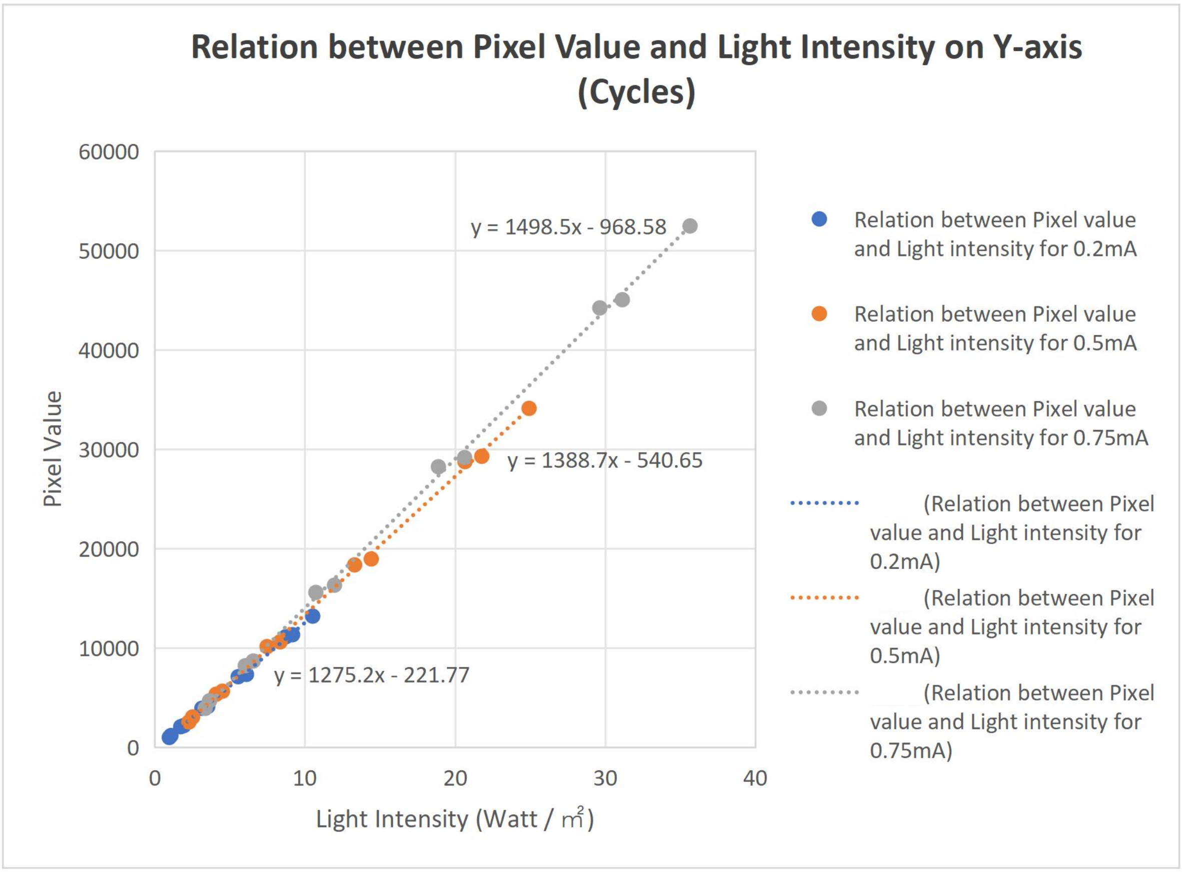


Figure 24. Relation between Pixel Value and Light Intensity on Y-axis

From the analysis of these four charts, there is a relatively obvious linear relationship between pixel value (Pixel Value) and illuminance (Lux) and light intensity (Watt/m²). Under different currents (0.2mA, 0.5mA, 0.75mA), the pixel value increases linearly with the increase of illuminance and light intensity, and the trends in the X-axis direction and the Y-axis direction are similar, indicating that the change trend of the pixel value in different axes is consistent. Then, the linear regression equations of the pixel value on illuminance and light intensity are drawn respectively through these scatter plots. The pixel value is linearly related to both illuminance and light intensity, indicating that the change of pixel value can be described by a linear model of illuminance or light intensity. And the data trends on the X-axis and Y-axis are the same, indicating that the linear relationship is established in different directions and is consistent.

6.2.3 Analysis for LuxCoreRender Engine

The first ones shown are the rendered images when LuxCoreRender is selected as the rendering engine (Figure 25, 26 and 27). The light source powers in the simulated environment are 0.53 mW, 1.37 mW and 2.1075 mW respectively.

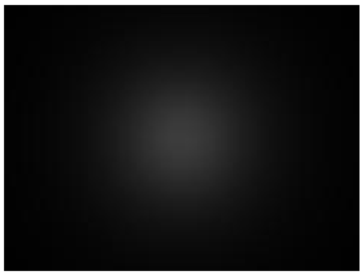


Figure 25. Rendering Light Distribution at 0.53 mW (LuxCoreRender)

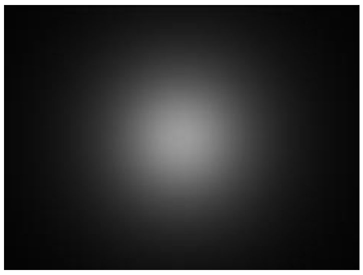


Figure 26. Rendering Light Distribution at 1.37 mW (LuxCoreRender)

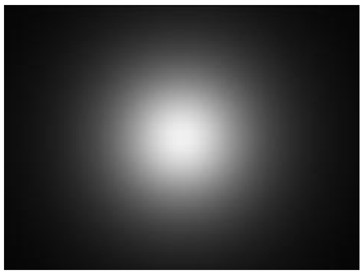
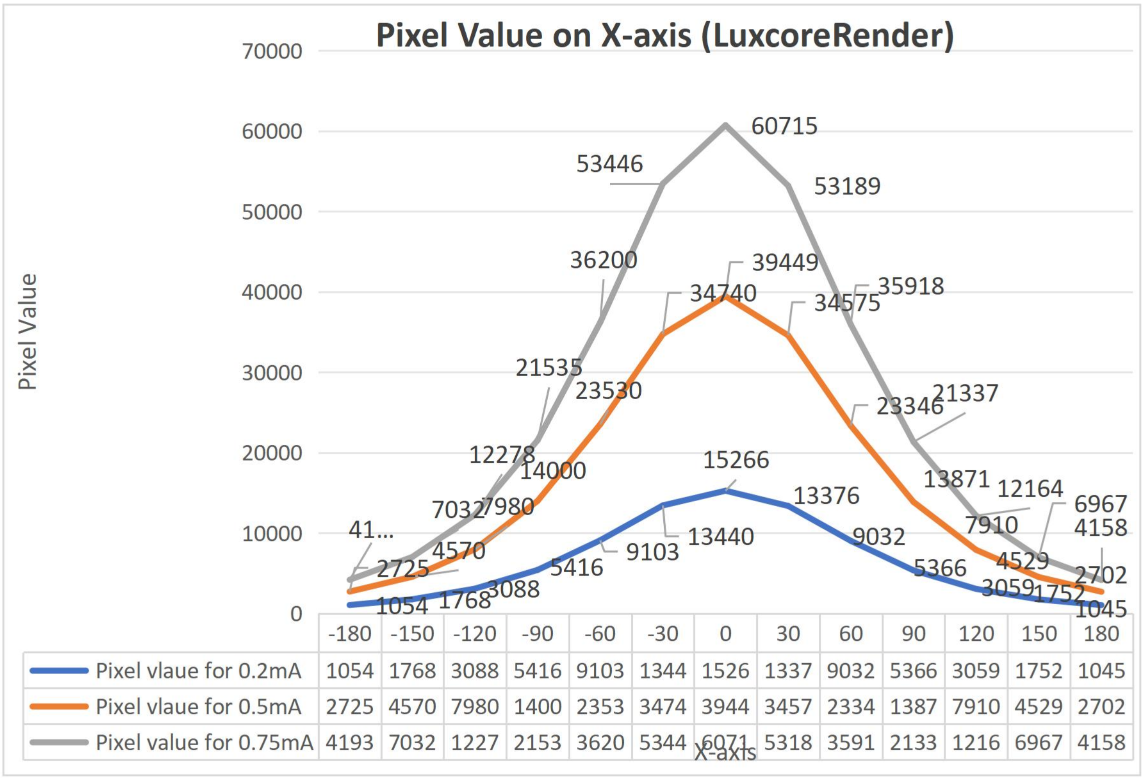


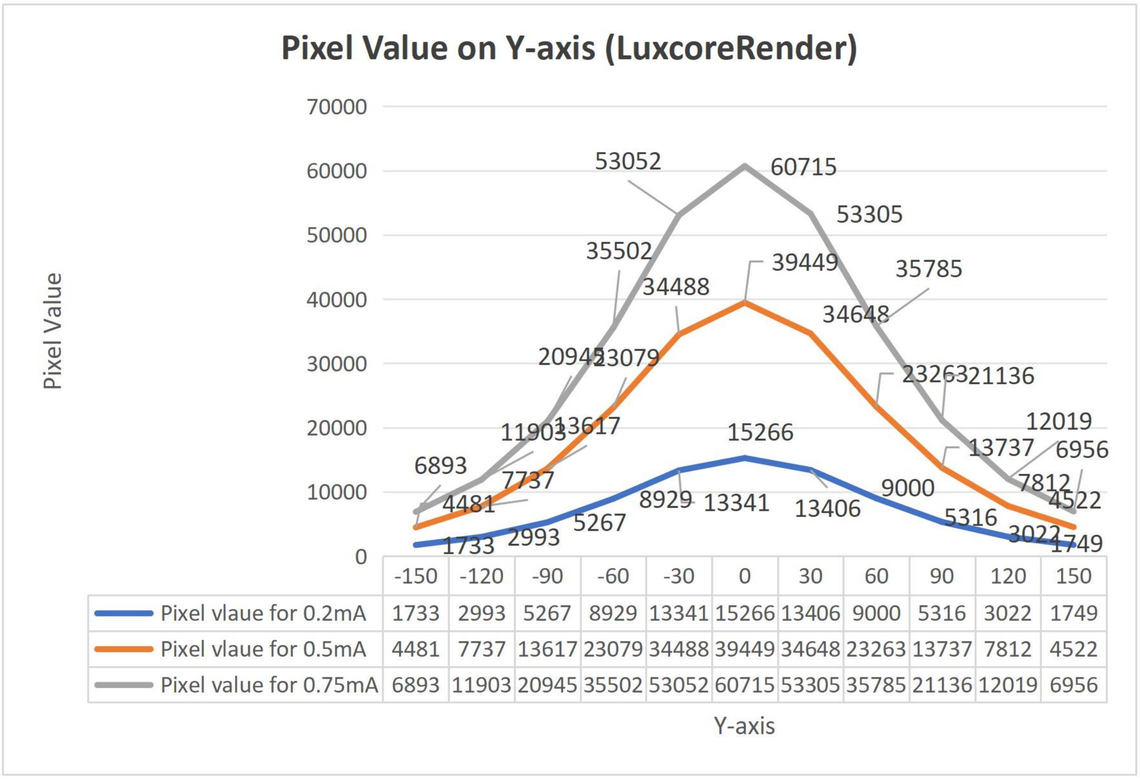
Figure 27. Rendering Light Distribution at 2.1075 mW (LuxCoreRender)

The following charts show the relationship between image pixel value and distance both on X-axis, Y-axis when using the LuxCoreRender engine.



X-axis	-180	-150	-120	-90	-60	-30	0	30	60	90	120	150	180
Pixel vlaue for 0.2mA	1054	1768	3088	5416	9103	1344	1526	1337	9032	5366	3059	1752	1045
Pixel vlaue for 0.5mA	2725	4570	7980	1400	2353	3474	3944	3457	2334	1387	7910	4529	2702
Pixel value for 0.75mA	4193	7032	1227	2153	3620	5344	6071	5318	3591	2133	1216	6967	4158

Figure 28. Pixel Value Distribution Along the X-Axis at 0.2 mA, 0.5 mA, and 0.75 mA (LuxCoreRender)



Y-axis	-150	-120	-90	-60	-30	0	30	60	90	120	150
Pixel vlaue for 0.2mA	1733	2993	5267	8929	13341	15266	13406	9000	5316	3022	1749
Pixel vlaue for 0.5mA	4481	7737	13617	23079	34488	39449	34648	23263	13737	7812	4522
Pixel value for 0.75mA	6893	11903	20945	35502	53052	60715	53305	35785	21136	12019	6956

Figure 29. Pixel Value Distribution Along the Y-Axis at 0.2 mA, 0.5 mA, and 0.75 mA (LuxCoreRender)

From these two charts, the vertical axis represents the pixel value and the horizontal axis represents the distance. The pixel value shows a symmetrical peak distribution with the change of distance, and the peak appears in the central area (0 position). Different currents (0.2mA, 0.5mA, 0.75mA) correspond to different curves. The larger the current, the higher the peak value, which means that higher current drive will make the light brighter, resulting in an increase in pixel value. The current value here corresponds to the current value used in the three tests in the actual physical model. It can also be observed that at positions far away from the center (±150 and ±180), the pixel value decays rapidly, and also shows the characteristics of a Gaussian distribution, which is similar to the relationship between illuminance, light intensity and distance in Figure 12, 13, 14 and 15.

Now, based on the actual illumination curve, actual light intensity curve, and the pixel value curve obtained by the LuxCoreRender engine, since their distribution forms are similar, it can be inferred that there is a linear relationship between them. The following four charts show the possible linear relationship between them (Figure 30, 31, 32 and 33).

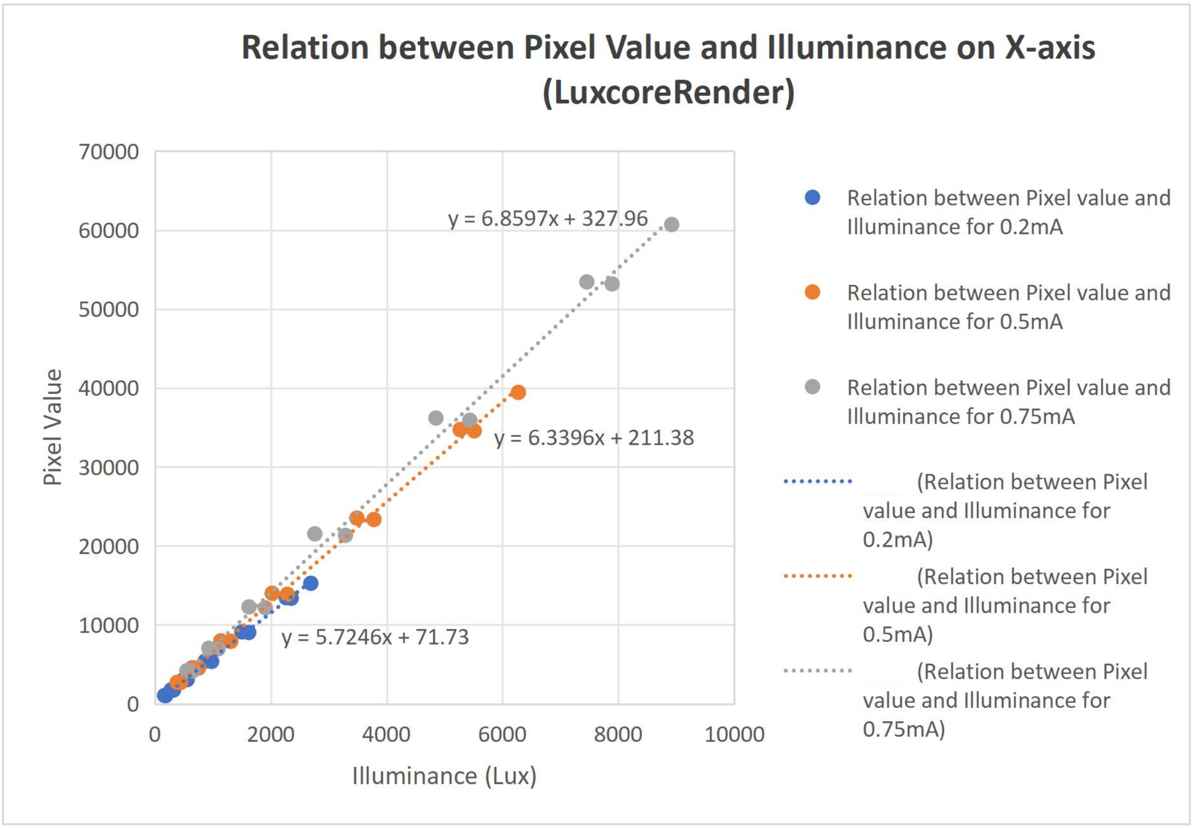


Figure 30. Relation between Pixel Value and Illuminance on X-axis (LuxCoreRender)

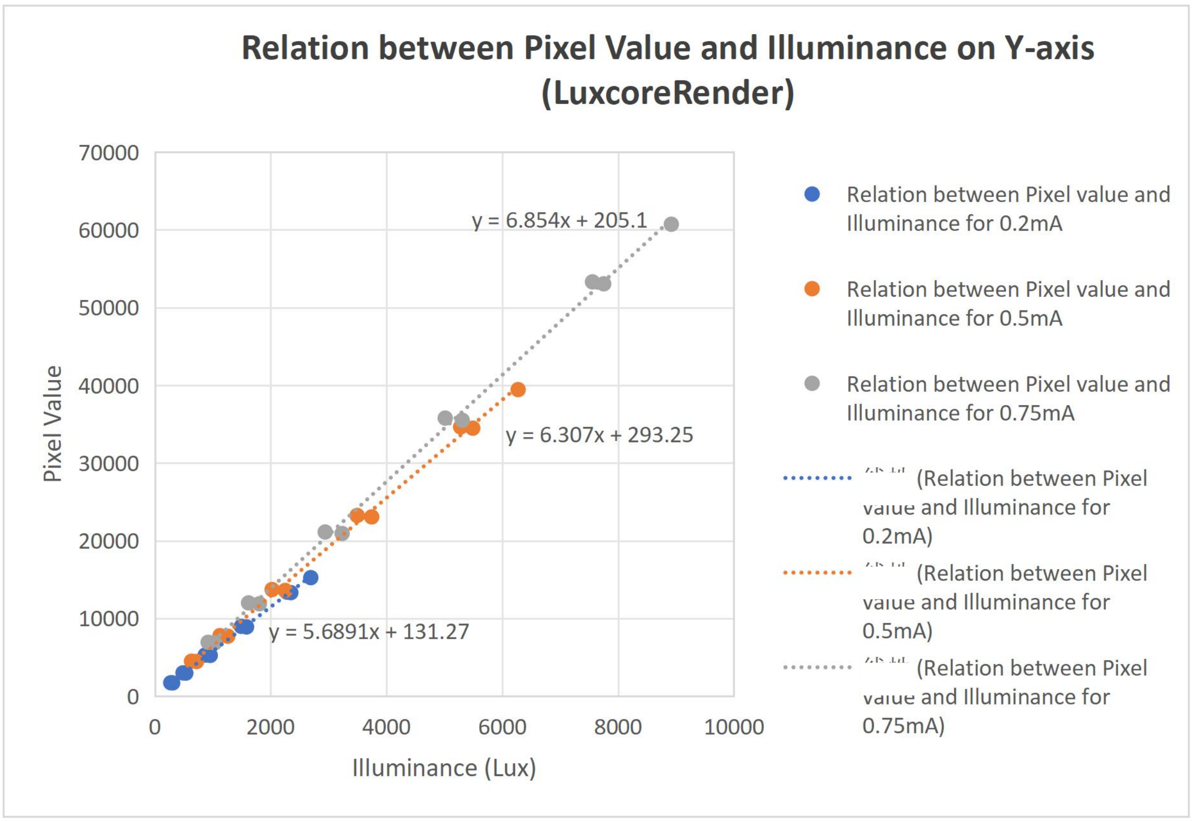


Figure 31. Relation between Pixel Value and Illuminance on Y-axis (LuxCoreRender)

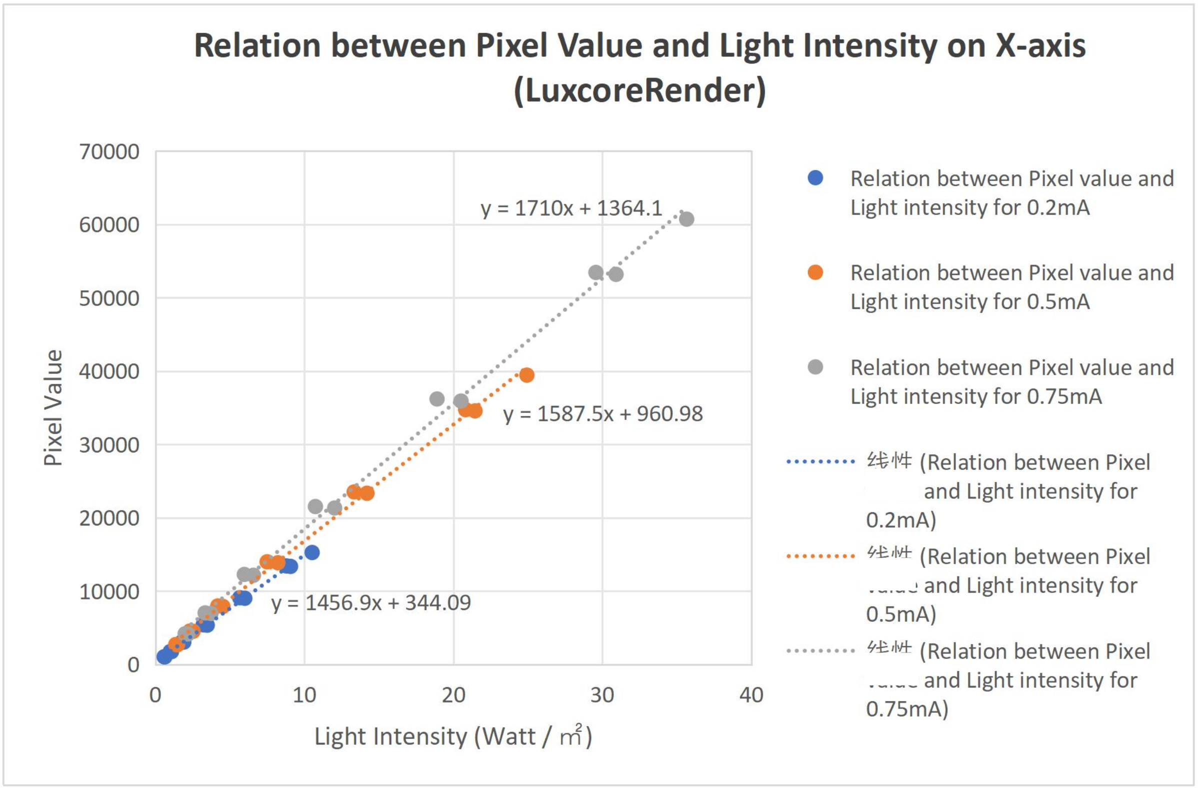


Figure 32. Relation between Pixel Value and Light Intensity on X-axis (LuxCoreRender)

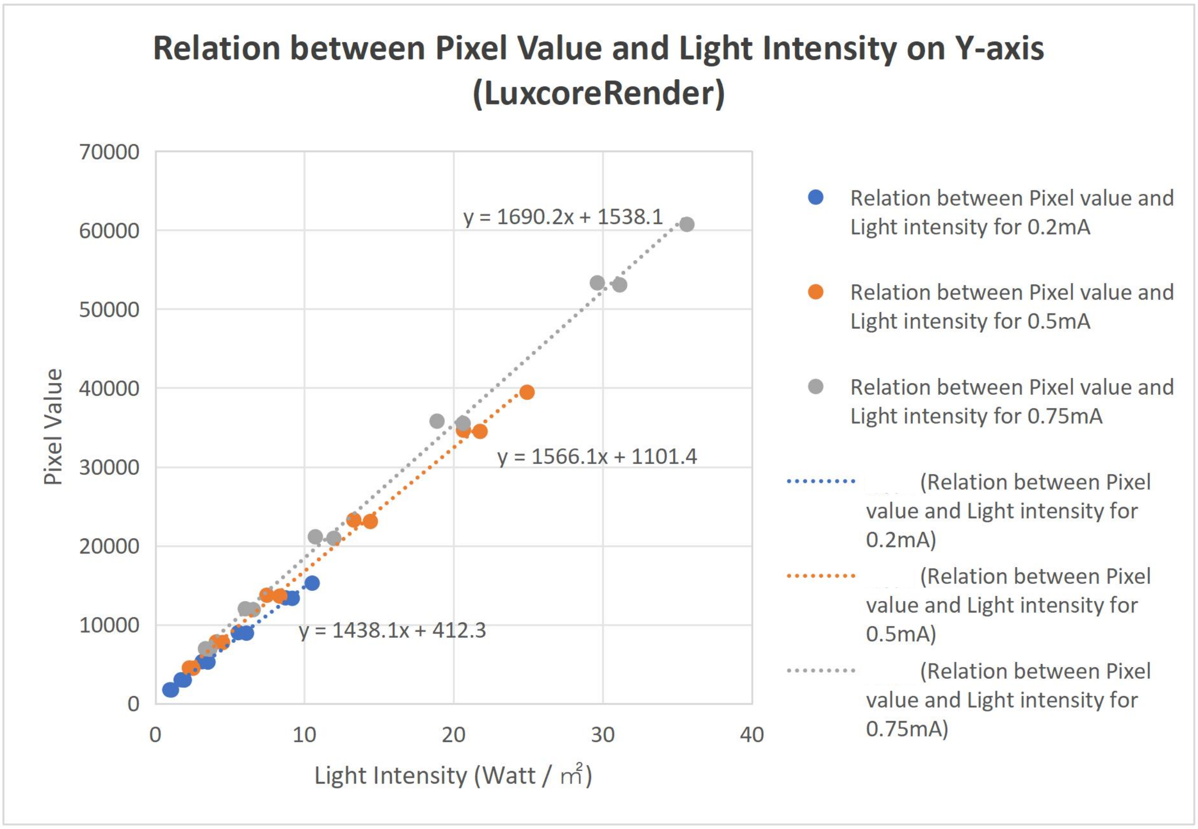


Figure 33. Relation between Pixel Value and Light Intensity on Y-axis (LuxCoreRender)

From the analysis of these four charts, there is a relatively obvious linear relationship between pixel value (Pixel Value) and illuminance (Lux) and light intensity (Watt/m²). Under different currents (0.2mA, 0.5mA, 0.75mA), the pixel value increases linearly with the increase of illuminance and light intensity, and the trends in the X-axis direction and the Y-axis direction are similar, indicating that the change trend of the pixel value in different axes is consistent. Then, the linear regression equations of the pixel value on illuminance and light intensity are drawn respectively through these scatter plots. The pixel value is linearly related to both illuminance and light intensity, indicating that the change of pixel value can be described by a linear model of illuminance or light intensity. And the data trends on the X-axis and Y-axis are the same, indicating that the linear relationship is established in different directions and is consistent.

6.2.4 Data Validation both for Cycles and LuxCoreRender

By analyzing the data obtained by the two engines, Cycles and LuxCoreRender, we can see that the change in pixel value can be described by a linear model of illuminance or light intensity. This also shows that after knowing the pixel value, the illuminance or light intensity in the actual situation can be inferred. In order to verify the feasibility of this method, the data obtained by the two engines are verified respectively. First of all, a linear regression model is established using the scatter plot data obtained when the simulated current is 0.2mA and 0.5mA. It can be implemented using Python. Figures 34 and 35 are from Cycles, Figures 36 and 37 are from LuxCoreRender.

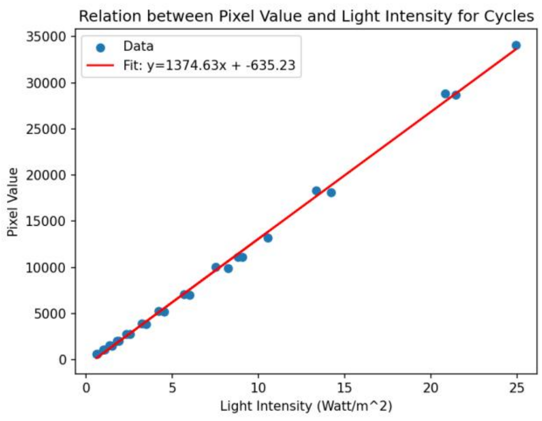
It can be seen that the linear equations of the four linear regressions are：

For Figure 34, $\mathrm{y} = 1374.63\mathrm{x} - 635.23 \quad (3)$

For Figure 35, $\mathrm{y} = 5.44\mathrm{x} - 1037.13 \quad (4)$

For Figure 36, $\mathrm{y} = 1606.79\mathrm{x} + 248.4 \quad (5)$

For Figure 37, $\mathrm{y} = 6.36\mathrm{x} - 241.37 \quad (6)$



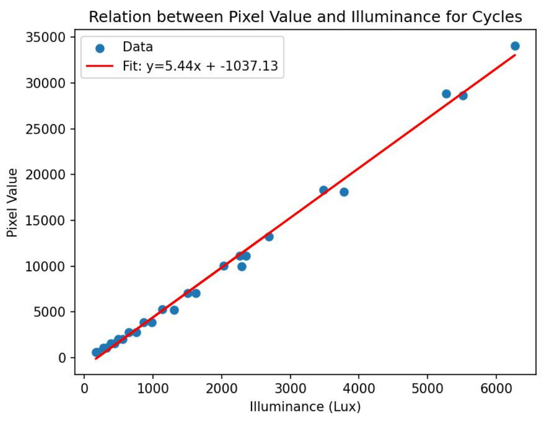
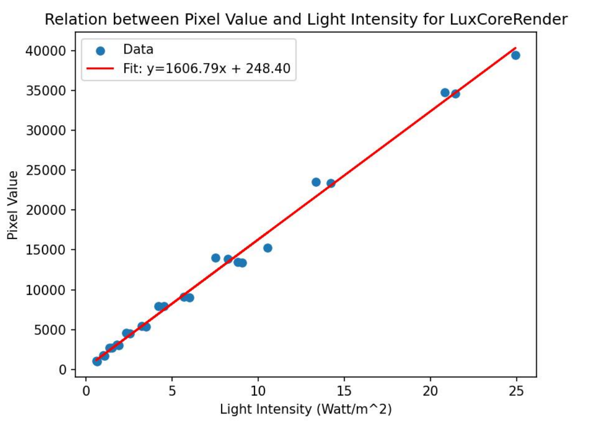


Figure 34. Pixel-Light Intensity Relation (Cycles)

Figure 35. Pixel-Illuminance Relation (Cycles)



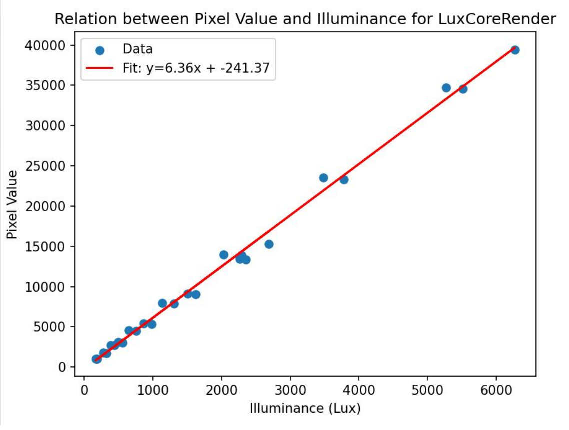


Figure 34. Pixel-Light Intensity Relation (LuxCoreRender)

Figure 35. Pixel-Illuminance Relation (LuxCoreRender)

Now use the four equations from (3) to (6) obtained to predict the illuminance and light intensity when the test current is 0.75mA, and then compare them with the actual values.

For Cycles engine, using equation (3) and (4) to predict light intensity and illuminance, then the following table can obtained. Table 4 shows the relationship among pixel value, predicted illuminance/light intensity and real illuminance/light intensity. The data shows that the illuminance is strongest at x=0 mm and gradually weakens to both sides, which is in line with expectations. The predicted value and the real value have the same trend, but there is a certain error. There are three possible sources of error. The first is the linear model error. The predicted illuminance may use linear fitting, while the real illuminance may have more complex nonlinear changes. The second is the device error. The device that measures the illuminance may have systematic errors or noise. The third is that the environment where the actual physical model is located cannot be completely consistent with the environment simulated by the virtual platform.

Coordinates			Pixel Vlaue (Cycles)	Predicted Light	Predicted Illuminance	Real Light Intensity	Real Illuminance
x(mm)	y(mm)	z(mm)					
-180	0	100	2367	2.18	625.76	2.00	552
-150	0	100	4301	3.59	981.27	3.34	930
-120	0	100	8109	6.36	1681.27	5.97	1624
-90	0	100	15453	11.70	3031.27	10.74	2760
-60	0	100	28156	20.94	5366.38	18.9	4851
-30	0	100	44333	32.71	8340.10	29.56	7455
0	0	100	52463	38.63	9834.58	35.65	8920
30	0	100	44093	32.54	8295.98	30.91	7891
60	0	100	27906	20.76	5320.43	20.51	5437
90	0	100	15287	11.58	3000.76	12.03	3290
120	0	100	8027	6.30	1666.20	6.58	1907
150	0	100	4257	3.56	973.19	3.73	1092
180	0	100	2344	2.17	621.53	2.18	650

Table 4. Relationship among Pixel Value, Predicted Illuminance/Light Intensity and Real Illuminance/Light Intensity (Cycles)

For LuxCoreRender engine, using equation (5) and (6) to predict light intensity and illuminance, then the following table can obtained. Table 5 can be observed that the illuminance and light intensity predicted by the LuxCoreRender engine also have errors. The reasons for this error are the same as those for the previous error when using the Cycles engine.

Coordinates			Pixel Vlaue (LuxcoreRender)	Predicted Light	Predicted Illuminance	Real Light Intensity	Real Illuminance
x(mm)	y(mm)	z(mm)					
-180	0	100	4193	2.45	697.23	2.00	552
-150	0	100	7032	4.22	1143.61	3.34	930
-120	0	100	12278	7.49	1968.45	5.97	1624
-90	0	100	21535	13.25	3423.96	10.74	2760
-60	0	100	36200	22.37	5729.78	18.9	4851
-30	0	100	53446	33.11	8441.41	29.56	7455
0	0	100	60715	37.63	9584.33	35.65	8920
30	0	100	53189	32.95	8401.00	30.91	7891
60	0	100	35918	22.20	5685.44	20.51	5437
90	0	100	21337	13.12	3392.83	12.03	3290
120	0	100	12164	7.42	1950.53	6.58	1907
150	0	100	6967	4.18	1133.39	3.73	1092
180	0	100	4158	2.43	691.72	2.18	650

Table 5. Relationship among Pixel Value, Predicted Illuminance/Light Intensity and Real Illuminance/Light Intensity (LuxCoreRender)

In summary, it is feasible to use the lighting simulation rendering functions of Cycles and LuxCoreRender in Blender to predict the real illuminance and real light intensity.

Future research needs to focus on model optimization, experimental optimization and error reduction.

7. Conclusion

This internship is an extremely valuable practical experience for me in the field of machine vision. It not only allows me to closely combine theoretical knowledge with practical applications, but also allows me to practice my ability of solving problems independently during the research and development process, and have a deeper understanding of the virtualization and simulation of machine vision systems. Through this internship, I have accumulated a wealth of experimental data, mastered professional skills, and also gained a deeper understanding of the development direction of industry technology.

Main research findings

During this internship, my main tasks revolved around illumination simulation and data analysis. Through a series of experiments and data analysis, I found that there is a certain mathematical relationship between pixel value and light intensity, which is of great significance for the accurate illumination simulation of machine vision systems. After a lot of data measurement and modeling, I successfully established a mathematical model that can predict the changes in pixel values under different lighting conditions and provide reliable theoretical support for subsequent optimization.

In addition, during this internship, I also discovered the key role of IES files in illumination simulation. By comparing the lighting distribution of different IES light sources, I confirmed the performance characteristics of different types of light sources in the virtual environment, and optimized the lighting parameters of the Blender Cycles/LuxCoreRender rendering engine to improve the accuracy of lighting simulation. This research result can provide valuable reference for industrial lighting simulation and promote the application of machine vision systems in complex lighting environments.

Contribution of the internship

This internship not only played a vital role in improving my academic research ability and practical skills, but also made certain contributions to the technical development of Genesi Elettronica. The following are the main contributions of this internship:

(1) Building an accurate virtual lighting system

During the internship, I successfully built a virtual lighting simulation platform based on the Blender Cycles rendering engine, and combined with IES files to achieve highly realistic lighting simulation. The system can simulate the characteristics of light sources in industrial machine vision systems and provide a reliable experimental basis for lighting optimization in different scenarios.

(2) Establishment of a mathematical model of pixel value and light intensity

Through systematic data measurement and analysis, I established a relationship model between pixel value and light intensity, and verified the applicability of the model under different experimental conditions. This research result not only helps to improve the accuracy of lighting simulation, but also provides mathematical support for the company's future research in machine vision.

(3) Optimization of light source parameters and lighting distribution

Through the analysis of IES data of different light source types, I optimized the angle, intensity distribution and attenuation characteristics of the light source, making the lighting simulation closer to the real environment. This optimization not only improves the physical accuracy of rendering, but also provides important practical experience for the design of lighting systems in the field of industrial automation.

(4) Rendering efficiency improvement and computing costs reduction

During the experiment, I used optimization techniques such as multiple importance sampling (MIS) and adaptive sampling to improve rendering efficiency and reduce computing resource consumption. This optimization enables the lighting simulation system to achieve higher computing efficiency while ensuring high-quality rendering.

(5) Reliable data support and experimental results

The data accumulated in this internship has been rigorously statistically analyzed and experimentally verified, and can provide high-quality data support for subsequent research. At the same time, I wrote a detailed technical report during the experiment, summarizing the research methods, experimental parameters and result analysis, and provided a reproducible experimental process for subsequent researchers.

Overall, the research results of this internship are not only of great value to personal skill growth and academic research, but also provide a useful exploration for Genesi Elettronica's technological development in the field of lighting simulation. This experience made me deeply realize the importance of combining theory with practice, and also made me more clear about the future research direction.

8. Bibliography

1. Kajiya, J. T. (1986, August). The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (pp. 143-150).
2. Pharr, M., Jakob, W., & Humphreys, G. (2023). *Physically based rendering: From theory to implementation*. MIT Press.
3. Jensen, H. W. (2001). *Realistic image synthesis using photon mapping*. AK Peters/crc Press..
4. Veach, E., & Guibas, L. (1995). Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques* (pp. 145-167). Berlin, Heidelberg: Springer Berlin Heidelberg.
5. LuxCoreRender. (n.d.). *LuxCoreRender – Physically based rendering engine*. https://luxcorerender.org/physically-based-rendering/
6. Keller, A. (1996). Quasi-monte carlo methods in computer graphics. *ZAMM-Zeitschrift fur Angewandte Mathematik und Mechanik*, *76*(3), 109-112.
7. Veach, E., & Guibas, L. J. (1997, August). Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (pp. 65-76).
8. Kelemen, C., Szirmay-Kalos, L., Antal, G., & Csonka, F. (2002, September). A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum* (Vol. 21, No. 3, pp. 531-540). Oxford, UK: Blackwell Publishing, Inc.
9. Cline, D., Talbot, J., & Egbert, P. (2005). Energy redistribution path tracing. *ACM Transactions on Graphics (TOG)*, *24*(3), 1186-1195.
10. Pantaleoni, J. (2017). Charted metropolis light transport. *ACM Transactions on Graphics (TOG)*, *36*(4), 1-14.

11. Walsh, J. W. (1959). The illuminating engineering society, 1909–1959. *Journal of the Institution of Electrical Engineers*, *5*(52), 221-223.

12. Blender Foundation. (n.d.). *Light settings*. Blender Manual. https://docs.blender.org/manual/en/latest/render/cycles/light_settings.html

13. Sadeghein, H. (2024). An overview and implementation of reservoir-based spatiotemporal importance resampling (ReSTIR) for real-time ray-tracing.

14. Hachisuka, T., & Jensen, H. W. (2009). Stochastic progressive photon mapping. In *ACM SIGGRAPH Asia 2009 papers* (pp. 1-8).

15. IESNA Testing Procedures Committee. (2008). IES Approved Method for the Electrical and Photometric Measurements of Solid-state Lighting Products. *Illuminating Engineering Society*.

16. Shirley, P. S. (1991). *Physically based lighting calculations for computer graphics*. University of Illinois at Urbana-Champaign.

17. Gadia, D., Lombardo, V., Maggiorini, D., & Natilla, A. (2024). Implementing Many-Lights Rendering with IES-Based Lights. *Applied Sciences*, *14*(3), 1022.

18. Blender Foundation. (n.d.). *Light object*. Blender Manual. https://docs.blender.org/manual/en/latest/render/lights/light_object.html

9. Appendix

Appendix A

```
import bpy
from PIL import Image

# FUNCTIONS #################################

def access_pixel_value(image_path, x, y):
    """
    Accesses the pixel value at a given coordinate in an image.

    Args:
        image_path (str): Path to the image file.
        x (int): X-coordinate of the pixel.
        y (int): Y-coordinate of the pixel.

    Returns:
        tuple: A tuple of pixel values (R, G, B) for RGB images.
    """

    try:
        with Image.open(image_path) as img:
            # Load the pixel access object for efficient pixel access
            pixels = img.load()
            pixel_value = pixels[x, y]
            return pixel_value
    except FileNotFoundError:
        print(f"Image file '{image_path}' not found.")
    except IndexError:
        print(f"Pixel coordinates ({x}, {y}) are out of bounds for the image.")

########################################

print('Setting parameters...')
```

```
# Set the render resolution (adjust as needed)
bpy.context.scene.render.resolution_x = 800
bpy.context.scene.render.resolution_y = 600

# Set the file format
bpy.context.scene.render.image_settings.file_format = 'PNG'
bpy.context.scene.render.image_settings.color_depth = '16'
bpy.context.scene.render.image_settings.color_mode = 'BW'

# Set the engine
bpy.context.scene.render.engine = 'CYCLES or LUXCORE'

# Set the parameters for luxcore
bpy.context.scene.luxcore.config.device = 'OCL'
bpy.context.object.data.luxcore.power = 0.53
bpy.context.scene.luxcore.halt.time = 300
bpy.context.scene.luxcore.denoiser.enabled = True
bpy.context.scene.world.luxcore.rgb_gain = (0, 0, 0)

# Set the parameters for cycle
bpy.context.object.data.energy = 0.53
bpy.context.scene.cycles.samples = 65536
bpy.context.scene.cycles.adaptive_threshold = 0.01
bpy.context.scene.cycles.preview_adaptive_threshold = 0.01

# Set the output path
image_path = "C:\\Files for Blender\\Rendering Image\\tempBW_1.png"
bpy.context.scene.render.filepath = image_path

print('Starting the render...')
# Start the render
bpy.ops.render.render(write_still=True)

print('Done!')

# Access to a pixel
val = access_pixel_value(image_path, 40, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 100, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 160, 300)
```

```
print('Value: ' + str(val))\
val = access_pixel_value(image_path, 220, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 280, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 340, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 460, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 520, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 580, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 640, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 700, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 760, 300)
print('Value: ' + str(val))
print("End for data on X_Axis")
val = access_pixel_value(image_path, 400, 1)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 60)
print('Value: ' + str(val))

val = access_pixel_value(image_path, 400, 120)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 180)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 240)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 300)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 360)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 420)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 480)
print('Value: ' + str(val))
```

```
val = access_pixel_value(image_path, 400, 540)
print('Value: ' + str(val))
val = access_pixel_value(image_path, 400, 599)
print('Value: ' + str(val))
print("End for data on Y_Axis")
```