UNIVERSITY OF CALIFORNIA SAN DIEGO

Computing Images with Diverse Illumination Effects

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Shilin Zhu

Committee in charge:

Professor Hao Su, Chair
Professor Xinyu Zhang, Co-Chair
Professor Henrik Wann Jensen
Professor David Kriegman
Professor Ravi Ramamoorthi
Professor Zhuowen Tu

2021

University of California San Diego

2021

DEDICATION

This Ph.D. dissertation is a gift to my beloved family, especially to my parents. They are bright stars lighting up the darkest night, shining my way.

This is also for all the people who came before me in the history of computer science. I am standing on their shoulders, listening to their wisdom, and following their guidance.

To readers who are the future of this field, I hope you find this dissertation inspirational, and never cease to push the human race forward.

EPIGRAPH

“It is not easier to make a good picture than it is to find a diamond or a pearl.”

Vincent van Gogh (1853-1890)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research supervisors, including Dr. Hao Su, Dr. Xinyu Zhang, and Dr. Ravi Ramamoorthi. As internationally respected scholars, they set a great example of seeking the truth of science. Their unstoppable passion for pushing the boundaries of human knowledge gives me the faith to leap over every single obstacle in my scientific career.

I also want to thank Dr. Henrik Wann Jensen and Dr. Mark Meyer, who have contributed significantly to my research projects through years of solid collaborations. They show me that we can conduct extraordinary work by gathering wisdom from a team of talented people.

I am deeply honored to have worked with researchers, engineers, and artists from Apple, Disney Research, Walt Disney Imagineering, Weta Digital VFX, and Pixar Animation Studios. They teach me how the industry brings academic research outcomes into real life and creates products to benefit millions of people. These experiences eventually direct me to start a new story at Pixar after my Ph.D. journey ends.

The Visual Computing Center at the University of California San Diego provides me with a friendly, collaborative, and professional environment for scientific research, and I am thankful to all the members of this center for creating such a wonderful experience. The memory at UCSD is a lifelong treasure of mine. I also want to thank Google Research for offering me a Ph.D. Fellowship and supporting my research for years.

As always, my family is the harbor of my soul, and I cannot express enough how much time I owe them since I started this five-year-long journey thousands of miles from home.

Chapter 2, in full, is a reprint of the material as it appears in Eurographics Symposium on Rendering 2020 and Computer Graphics Forum 2020 [212]. Shilin Zhu, Zexiang Xu, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi, Wiley-Blackwell, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of two materials as they appear in ACM Transactions on Graphics 2021 [214, 215]. Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi, Association for Computing Machinery, 2021. The dissertation author was the primary investigator and author of these two papers.

Chapter 4, in full, is a reprint of the material as it appears in International Conference on Mobile Computing and Networking 2017 and Communications of the ACM 2020 [210, 213]. Shilin Zhu, Chi Zhang, and Xinyu Zhang, Association for Computing Machinery, 2020. The dissertation author was the co-primary investigator and author of this paper. Chi Zhang was the other co-primary author who contributed equally to this work.

Chapter 1 contains citations to other co-authored materials where I participated and contributed, and I am more than grateful for all the efforts made by my remarkable colleagues.

VITA

2012-2016	Bachelor of Science, University of Science and Technology of China
2016–2017	Research Assistant, University of Wisconsin, Madison
2017–2021	Google Ph.D. Fellowship Researcher, University of California San Diego
2018	Research and Development Intern, Apple Inc.
2019	Master of Science, University of California San Diego
2019	Research and Development Intern, Disney Research and Walt Disney Imagineering
2021	Rendering Intern, Weta Digital VFX
2021	Research Intern, Pixar Animation Studios
2021	Doctor of Philosophy, University of California San Diego
2022-	Research Scientist, Pixar Animation Studios

PUBLICATIONS

[1]"Deep Kernel Density Estimation for Photon Mapping." In Computer Graphics Forum, vol. 39, no. 4, pp. 35-45, Wiley-Blackwell, 2020.
[2]"Photon-Driven Neural Reconstruction for Path Guiding" ACM Transactions on Graphics (TOG) 41, no. 1 (2021): 1-15.
[3]"Hierarchical Neural Reconstruction for Path Guiding Using Hybrid Path and Photon Samples" ACM Transactions on Graphics (TOG) 40, no. 4 (2021): 1-16.
[4]"Survey: Machine Learning in Production Rendering." arXiv preprint arXiv:2005.12518 (2020).
[5]"Robust Multimodal Vehicle Detection in Foggy Weather Using Complementary Lidar and Radar Signals." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 444-453. 2021.
[6]"Deep Stereo using Adaptive Thin Volume Representation with Uncertainty Awareness." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2524-2534. 2020.
[7]"PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 909-918. 2019.
[8]"Automating Visual Privacy Protection using a Smart LED." In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, pp. 329-342. 2017.
[9]"Automating Visual Privacy Protection using a Smart LED." Communications of the ACM, 63, no. 2 (2020): 81-89.

[10]“Enabling High-Precision Visible Light Localization in Today’s Buildings.” In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, pp. 96-108. 2017.
[11]“Invisible QR Code Hijacking Using Smart LED.” Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies vol. 3, no. 3 (2019): 1-23.
[12]“Gait Recognition for Co-Existing Multiple People using Millimeter Wave Sensing.” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 1, pp. 849-856. 2020.
[13]“Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit?.” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4923-4932. 2019.
[14]“Towards Fast and Energy-Efficient Binarized Neural Network Inference on FPGA.” In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 306-306. 2019.
[15]“SimBNN: A Similarity-Aware Binarized Neural Network Acceleration Framework.” In 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 319-319. IEEE, 2019.

FIELDS OF STUDY

Major Field: Computer Science (Specialization in vision and graphics)

Studies in 3D Computer Vision and Machine Intelligence
Professor Hao Su

Studies in Mobile and Ubiquitous Computing
Professor Xinyu Zhang

Studies in Computer Graphics
Professors Ravi Ramamoorthi and Henrik Wann Jensen

ABSTRACT OF THE DISSERTATION

Computing Images with Diverse Illumination Effects

by

Shilin Zhu

Doctor of Philosophy in Computer Science

University of California San Diego, 2021

Professor Hao Su, Chair
Professor Xinyu Zhang, Co-Chair

Producing and capturing *images* with anticipated looks under various *illuminations* is the ultimate goal of computer graphics and computational photography. Two long-standing problems are blocking the pathway: how to generate pixels to form realistic appearances (image synthesis or rendering) and how to steer pixels to manipulate captured appearances (computational imaging and illumination). In this dissertation, both topics will be covered and discussed as part of the image formation task.

Physically-based rendering creates photo-realistic images by calculating pixel colors from light transport simulation, which has been used by the gaming and film-making industry for years to produce astounding visual effects on screens. However, synthesizing the entire family of illumination effects on the image is computationally expensive. It can readily cost days of time to yield a single frame in the existing production pipeline, especially when lights are challenging to compute by the standard ray tracing algorithm. To remedy the issue, we have proposed multiple methods to accelerate the sampling and reconstruction of different types of illumination, leading to more efficient Computer-Generated Imagery (CGI).

Computational illumination and digital imaging expand the function set of image synthesis in graphics by supporting more flexible appearance alterations during image construction. One of the unexplored functionalities is to adjust pixels structurally through unique patterns carried by coded illuminations. We have developed a dedicated lighting and camera system to collaboratively direct the captured image appearances, enabling unconventional lighting effects such as the selective restructuring of illuminated image segments in a controllable and automatic mode.

From the simulated light transport to real-world computational photography, we have advanced the visual computing technology by producing images with desirable looks and diverse illumination effects. Our conducted research also open up new challenges and opportunities for future studies on images.

Chapter 1

Introduction

Images are the center of almost every visual computing technology, such as computer vision, computer graphics, and computational photography. Unlike humans to perceive images as highly abstracted semantics and global structures, machines treat them simply as a discrete set of digital brightness values (i.e., pixels). Therefore, it has been one of the major missions of computer science to create sensible images that meet people's expectations under various illumination settings by computational algorithms, which is the central problem to study in this dissertation. In general, there are two categories to explore; One of which is to synthesize realistic appearances that coincide with the physics of light transport (Chapter 2 and Chapter 3), and the other is to control the appearance by manipulating the standard image formation process (Chapter 4).

It is a well-known fact that images are significantly affected by the lighting distributions in the scene to be photographed. In the following chapters, we investigate multiple types of illumination effects, such as concentrated caustics (Chapter 2), global lighting (Chapter 3), and coded light source (Chapter 4). Computing these diverse illuminating phenomena in the image requires a unique perspective and design for every one of them.

This chapter introduces an overview of imaging and lighting, followed by the basic concepts of graphical rendering and computational photography systems. Then, we take a brief tour of the published articles covered by the rest of the chapters and technical challenges in building computer algorithms for a variety of image formation tasks.

1.1 Motivation

1.1.1 Significance of Images

Image is an essential ingredient for understanding our 3D world. A massive amount of pictures are created by people every day to record their precious moments on mobile devices, by entertainment studios like Disney for presenting immersive visual effects on the computer or theatre screens, and by robots to adapt themselves when performing challenging tasks such as the Mars Rovers by Jet Propulsion Laboratory (JPL). We, as livings, also capture sequences of images with our naked eyes to assist various interactions with the environment. Without advanced imaging functionalities, modern civilizations may not even be able to survive.

It is the mission of computer science to study the principle of imaging and understand how appearance is formed when photographing a scene with diverse lighting. In the past decades, researchers and engineers in this field have developed a large number of computational algorithms to create images of expectation in both virtual and real setups. Such rapid development leads to a wide range of influential visual applications such as virtual characters and environments, movie effects like explosions and space travel, powerful cameras to trace individual photons, and giant telescopes for picturing galaxies.

1.1.2 A Brief History of Making Pictures

The study of classic optical imaging can be traced back to ancient times where pinhole cameras and 3D to 2D projections were presented as early as in Chinese philosopher *Mozi* writings (circa 400 BCE), *The Optics* (Latin: De aspectibus or Perspectivae) by Arab physicist Ibn al-Haytham (965-1040), among many other manuscripts. These ancient image formation and geometric optics theories have been known to inspire many great scientists and artists such as René Descartes, Johannes Kepler, and Leonardo Da Vinci. Although complicated modern imaging techniques have replaced these original computing models, their findings are the foundations and ancestors of everything we have today.

Before digital imaging was invented, the image was formed on photographic film pioneered by George Eastman since 1885, who founded *Kodak*, which was later applied to motion pictures by the movie industry. After the invention of metal oxide semiconductor (MOS) technology by engineers at *Bell Labs* in 1959, the digital semiconductor image sensor became foreseeable, leading to charge coupled device (CCD) in 1969 and CMOS sensor in 1993 subsequently. There were precursor developments in electronic displays such as the Ivan Sutherland's Sketchpad on MIT TX-2 computer with cathode-ray tubes (CRT) in 1962 and cinematography that can be dated back to the 1900s when Lumiere brothers invented the early visual effects before the birth of computer-centered imaging.

The term computer graphics was coined in the 1960s, and the world's first ray casting light transport algorithm was presented by Arthur Appel in 1968 to target photorealism in image synthesis. In the 1970s, thanks to the emergence of MOS integrated circuit chips, computer graphics became more practical with faster hardware, and people started to produce raster-based images (e.g., SuperPaint) from 1972. At that time, The legendary visual effects studio - *Lucasfilm* was founded. Subsequently, Ivan Sutherland and David C. Evans at the University of Utah started to instruct computer graphics, and the emblematic Utah teapot image was created. One of the students, Edwin Catmull, the co-founder of *Pixar*, invented several milestone CGI techniques, including texture mapping on 3D models in 1974, and he later contributed significantly to Pixar's RenderMan. In the early 1980s, central processing unit (CPU) microprocessors and the early graphics processing unit (GPU) chips started to revolutionize computer graphics. Afterward, the general rendering equation for computing global illumination on images was derived by James Kajiya in 1986, which has become the central piece of forming an image solely by computers. After two years, Pixar developed the first shader programs in 1988, followed by producing the first fully computer-generated 3D motion pictures - *Toy Story* in 1995. From the 1990s to 2010s, the photorealism of images continued to improve by inventing new image reconstruction

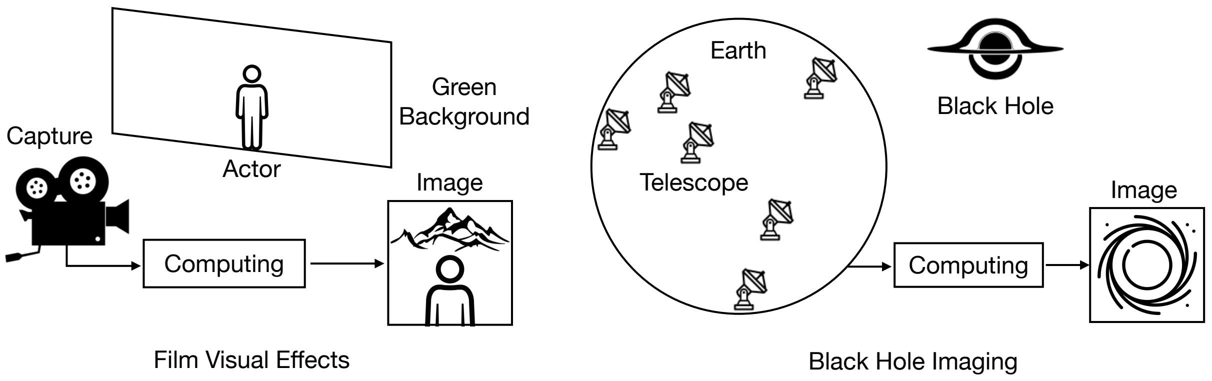


Figure 1.1. Example applications of image formation by computations. *Left:* Computer generates background effects such as mountains and rivers for film actors by photo-realistic rendering technology. *Right:* Multiple telescopes collaboratively form an image of black hole by computational photography techniques.

algorithms such as bidirectional ray casting and particle-based rendering, and CGI started to thrive in the industry. The physically-based rendering (PBR) became widely popularized by Matt Pharr, Greg Humphreys, and Pat Hanrahan in 2014. Today, images produced by computers with robust and efficient light transport simulations are ubiquitous in our daily lives.

Compared to the long history of classic cameras and computer graphics, computational photography with programmable illumination and camera is still a rising young field. Unlike traditional ways of making pictures solely from optical processes and photographic film, a significant amount of digital computing is involved in this new type of photography. Therefore, this is achievable only after the 1970s when the solid-state image detector was invented, such as the development of high dynamic range (HDR) imaging by Charles Wyckoff in the early 1980s. It took about 25 years for the digital sensor to be reliable and cost-effective, and combined with the sustained rise of computing powers of CPU and GPU since the 1980s, computational photography started to become practical and less bulky from the early 2000s. Through this short development period, there has been a considerable amount of achievements made by scientists, including Steve Mann, Marc Levoy, Shree K. Nayar, and Ramesh Raskar. Their work gave a much broader meaning to this field by extending and enhancing the capabilities of digital imaging and producing pictures that were considered impossible by traditional photography. Until today, some representative examples are light field camera, telescope for capturing black holes, trillion FPS camera, HDR and panoramic imaging, effects such as mosaicing and watermarking, coded aperture and shutter, optical tomography, holographic imaging, specialized retina, dual imaging, lighting domes, 4D illumination, active encoded lights, and multi-flash imaging. Starting from the 2010s, machine learning took off, and it has been benefited a lot by having a rich-featured input from this new means of capturing such as encoded polarization images. The invention of producing images by computation over optical projection is a remarkable moment in the history of photography.

Humans have spent a long time investigating the ways of making pictures ever since the sun lights the earth. From ancient pinhole observations to classic optics and from photographic film to digital sensors, technology has revolutionized the way of perceiving the 3D world. Today, we are able to produce realistic images from computers by graphical rendering and enhance the imaging capabilities by highly performant digital computations, as shown in Figure 1.1. Many visual fantasies from science fictions are gradually becoming facts such as virtual and augmented reality. These are all thanks to the great efforts made by people who came before us.

1.2 Relations Between 2D Images and 3D World

Having reviewed the significance and history of imaging, we introduce the high-level concepts of the image formation process in this section. There are many components involved in performing a 2D screenshot of the 3D space, including camera, lighting, scene geometry, and material. Understanding how the image appearance relates to these components is critical for building light transport and photography algorithms to produce pixels with the desirable look.

1.2.1 Fundamentals of Standard Imaging Process

Photographing a 3D scene requires a few steps and is affected by multiple components, as shown in Figure 1.2. To begin with, imaging is a process of gathering energy transmitted to the image receptor on the camera. Light sources send out photons in visible spectrum controlled

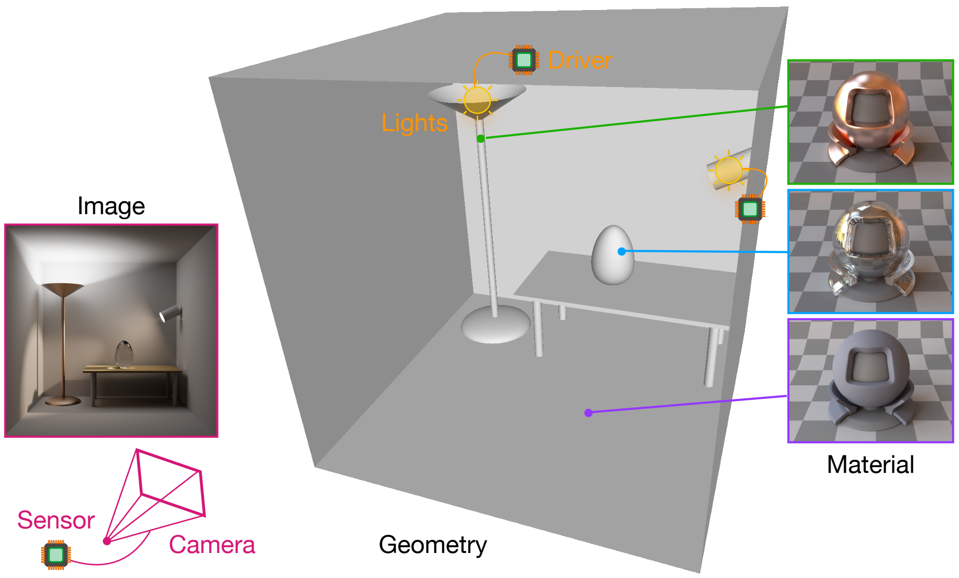


Figure 1.2. Basic components of forming an image. In this example scene, a camera with an image sensor captures the incoming energy transmitted towards it for each pixel to form an image. A driver can control the light source to change its illuminating patterns. The 3D scene geometry causes the light rays to bounce in between different objects that have different material properties, such as the diffuse wall, the glass egg, and the metal pole in this case.

by the driver circuit, illuminating the 3D scene of interest. Next, we have a 3D geometric representation (e.g., polygonal meshes) of the scene to be photographed, where materials and textures are defined on the object surfaces determining their unique responses to the incident illumination. By combining these small pieces together, we can compute the global radiance distribution within the scene and generate color appearances on the final image.

As people may observe, in the standard imaging process, the electronics are placed at two terminal devices (i.e., camera and light source), while the energy transport between them is guided by the physics of optics. Computer graphics mainly focuses on simulating the light transmission by physically-based rendering, and computational photography adds new features to illumination patterns and camera sensors. Despite the difference, the target of both fields is the same - computing the final picture.

While the global light field in a scene is distributed continuously in 3D space, we can only acquire a discrete representation of the continuous illumination signals for digital imaging

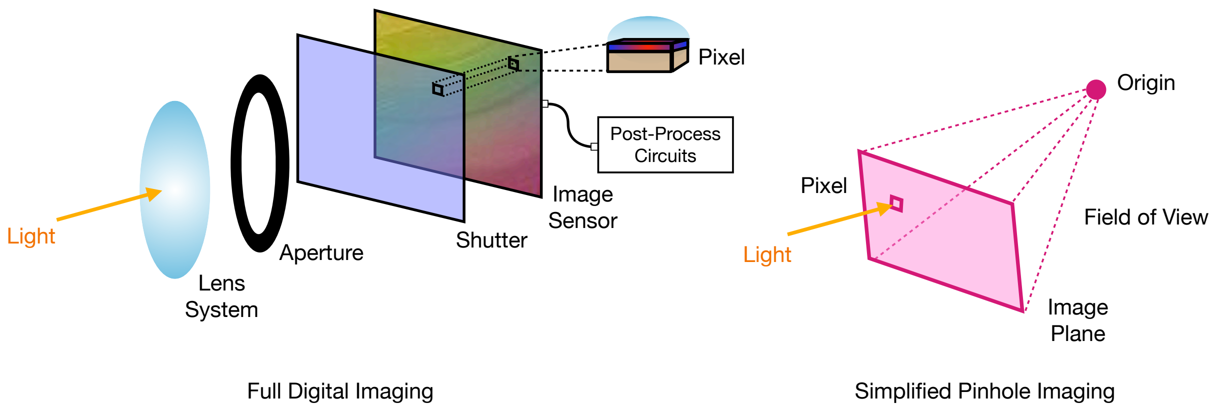


Figure 1.3. Illustration of camera models in digital imaging. *Left:* The cross section of a real camera. The incident light travels through a few optical and digital devices before hitting the photon receptors on the sensor. *Right:* The simplified pinhole model is used for computing images when some camera parts are reasonably neglected.

with finite-resolution rasterized sensors. Despite such limitation compared to photographic film, discretization offers the opportunity to leverage computational algorithms to form an image, which can sometimes be extremely difficult for analog devices. Therefore, in this dissertation, we only consider digital photography where computers participate heavily in the loop.

1.2.2 Camera and Lighting Basics

With the fundamental concepts in mind, we can review the technical details involved in the process of imaging, including the structure of camera, the principle of illumination, and the emergence of appearance. The formal mathematical definitions will be covered in Chapter 2, Chapter 3, and Chapter 4 correspondingly. Since we mainly consider the realm of digital photography, the continuous illumination distribution will be converted into discrete and numerically computable forms when we design specific algorithms.

Camera Model and Digital Image. Inspired by the laws of optics, modern cameras are designed to contain both optical components and digital circuits for maximizing imaging efficiency. As shown in Figure 1.3 (left), the compound lens and aperture mechanics control the direction and amount of light that can pass into the camera, followed by the digital shutter to manage the exposure time of individual pixels installed on the backend image sensor. The semiconductor sensor consists of a 2D array of pixels with a color filtering layer to produce red, green, and blue (RGB) channels of an image. Finally, the data is streamed to post-processing circuits for rectification. Therefore, the appearance of the final image is determined by various camera parts, where every single piece can be driven by a separate computational algorithm.

In this dissertation, we put our primary focus on the electronic shutter and image sensor while assuming the lens and aperture are properly configured already. Typically, there are two kinds of shutter technology. The global shutter enables the entire sensor to receive the outside illumination simultaneously, while the rolling shutter schedules the exposure from top to bottom rows asynchronously. Pixels on the sensor must be assigned sufficient integration time to receive a reasonable amount of photons for computing brightness values. Such control flexibility offers the opportunity to establish novel image appearances with programmable shutters. Chapter 4 presents an example of leveraging a computational algorithm to switch pixels on and off adaptively.

Following a different philosophy, the camera model can be greatly simplified in particular cases while only keeping the parts of interest intact. Obviously, some phenomena cannot be captured and will be lost by the approximation; however, the problem is insignificant when those effects are out of consideration for specific tasks. For instance, the pinhole model in Figure 1.3 (right) is widely used by image synthesis in computer graphics. In this reduced model, a (virtual) image plane is placed in front of the origin representing the center of the camera, and their geometric connection determines the field of view (FoV). Scenes outside of the horizontal and vertical angle of view are invisible to the camera.

The above simplification makes the image formation easier to work with mathematically in rendering such as the ray tracing algorithm, as shown in Figure 1.4 (left). Camera rays are spawned from the origin and travel through the image plane to enter the 3D scene. When a ray hits a surface region illuminated by the light sources, we compute the amount of energy that can be transmitted back to the pixels on the image. Thus, adopting the simplified representation of the camera makes the image formation process more elegant to compute.

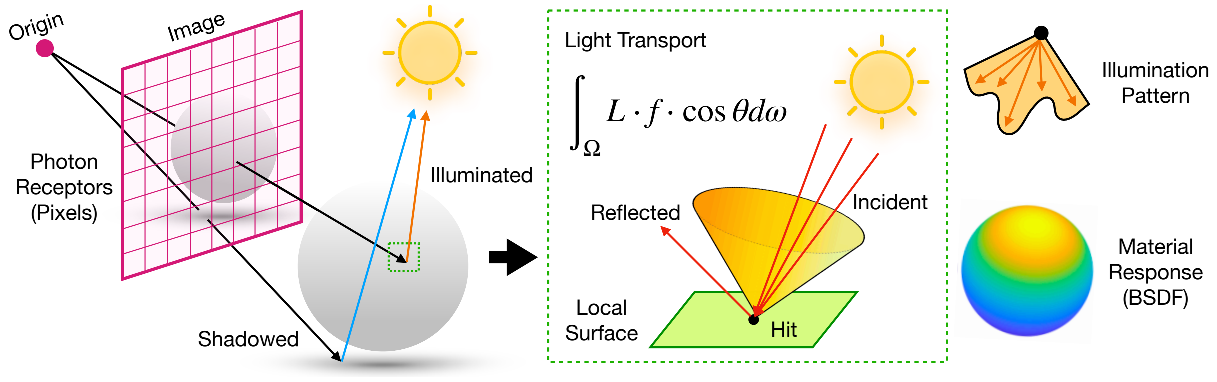


Figure 1.4. Principle of illumination and appearance. *Left:* Multiple light paths are constructed between camera and light source, generating different effects such as direct lighting and shadows. *Middle:* The bouncing of light on the surface is guided by the incoming radiance L, the material response f, and geometric relations between rays and local surface. *Right:* The surface appearance is controlled by the illumination and material distributions which are both editable.

Lighting and Appearance. An image is no more than a 2D lattice slice of the 3D world from the camera perspective. However, if we stand on the geometric surfaces or the light sources, imaging is a process of transporting photons through the space until reaching a pixel on the view, as illustrated in Figure 1.4. One prominent challenge of computing the overall pixel color is to discover all the probable transmission paths. In Figure 1.4 (left), we demonstrate the examples of the standard direct lighting and shadowing with a single bounce. Some special illumination effects are difficult to create, for example caustics (Chapter 2) and indirect lighting (Chapter 3), since their paths are troublesome to find.

To explain the principle of surface appearance, we study a unitary local interaction with the incident light as presented in Figure 1.4 (middle). Basically, the reflected radiance depends on the incoming radiance from all directions over the entire upper hemisphere centered at the hit point. Furthermore, the surface material modulates the reflected light, which is manifested by a spherical bidirectional scattering distribution function (BSDF). For instance, diffuse surfaces like walls scatter energy uniformly to all directions, while specular surfaces like mirrors only scatter exclusively to particular directions. Combining with additional geometric relations, the outgoing radiance is computed iteratively for every interaction (i.e., bounce) until reaching the image

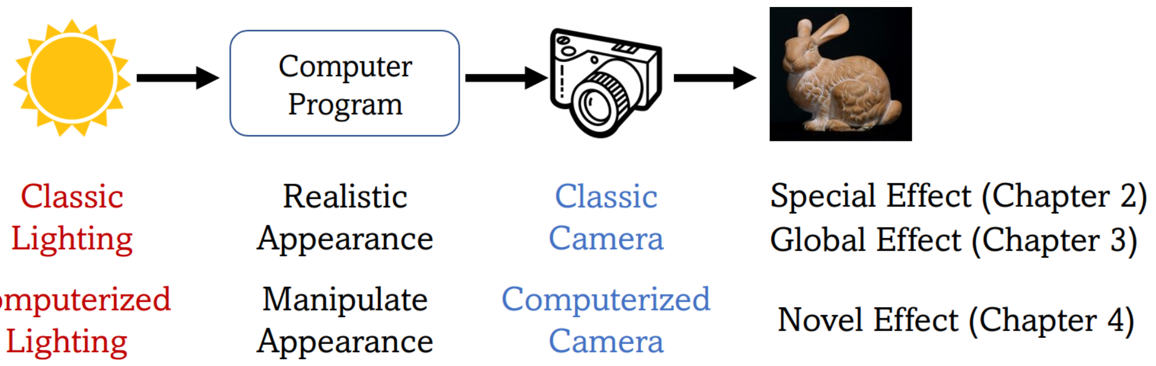


Figure 1.5. Illustration of the task space in this dissertation. Based on illumination and camera types, we build different computer programs to achieve diverse effects on images.

plane. Simulating light transport through paths and interactions to form a colored appearance is the central piece of image synthesis.

We strive to create images with diverse lighting effects so that we are able to govern the look. Some of these effects are only achievable by altering the illumination profile, such as the example shown in Figure 1.4 (right). Generally speaking, the illumination is programmable with computational algorithms to produce varying patterns with respect to space and time. Physically-based rendering normally applies the common light sources with constant brightness to imitate real-world conditions. In contrast, computational photography frequently utilizes active encoded lights and textured patterns to create dynamic variations. In Chapter 4, we present a paradigm of manipulating image appearances using computational illumination techniques.

To summarize, other than the viewpoint defined by the camera settings, the illumination and light transport majorly determine the final color of each pixel, yielding accurate appearances of scenes to be photographed. The complete imaging system grants us the maximum flexibility to form pictures.

1.2.3 Task Space of Image Computation

Computing images involves a family of algorithms to be applied for a variety of visual applications. There are many diverse pathways to categorize the tasks into separate spaces, and we consider two particular types of clustering in this dissertation (as shown in Figure 1.5):

- **Based on Visual Effects.** People constantly generate images with certain appearance expectations. Our conducted research work can be divided into distinctive cases - computing sharp caustics (Chapter 2), global illumination (Chapter 3), and dynamic lighting (Chapter 4). These effects are fairly inefficient to create using traditional methods, and each of them requires a customized approach to address its exclusive challenges.
- **Based on Applied Operations.** From the application perspective, the goal of imaging can be rather unique for different tasks. Synthesizing lifelike images (Chapter 2 and Chapter 3) is the fundamental intention when producing immersive games and films, which operates on light transport paths. On the contrary, if we shift the focus to two end components - camera and light source, we can innovate unconventional image appearances (Chapter 4) by computational photography algorithms.

To develop solutions for the above tasks, we follow the traditions of system implementation and evaluation. Rendering is performed in a synthetic world created by the computer program, and computational imaging is tested on authentic prototypes in the real world.

1.3 Challenges and Opportunities in Image Formation

The aforementioned image computation tasks are practically non-trivial from the algorithm design and implementation perspectives. Thus, we briefly recap the obstacles encountered and progresses accomplished by researchers in the past few years.

1.3.1 Monte-Carlo Light Transport for Image Photorealism

Realistic image synthesis is known as a long-established problem to conquer by the computer graphics community. Ever since the mathematical formulation - the rendering equation was found, people have made tons of attempts to comprehend and simulate the physics of light.

Historically, physically-based rendering did not receive much attention due to the lack of capability to carry out complex geometry processing, coherent memory footprint, and low-cost shading. At that moment, people adopted a micropolygon-based workflow and approximate various illumination effects using point clouds. Despite the advantage of possessing less run time and higher flexibility, such a method requires convoluted data handling, and diverse lighting effects cannot be computed in a unified routine. Consequently, the old-fashioned way was abandoned and subsequently replaced by Monte-Carlo (MC) path tracing, an elegant numeric solution that launches rays and simulates real-world light scattering events.

Unfortunately, the brute-force ray tracing suffers from a slow convergence problem due to the nature of Monte Carlo, producing images with significant noise and defective effects. As opposed to spawning a huge number of rays, people have unfolded a variety of variance reduction techniques to expedite the convergence, such as importance sampling, multi-directional path tracing, particle-based rendering, path guiding, and denoising. However, even with these state-of-the-art light transport algorithms, the space for improvement is still quite substantial because of the unceasing growth of geometric and lighting complexity.

1.3.2 Programmable Photography for Image Manipulation

The idea of unlocking the full potential of digital photography has promoted the emergence of special-purpose cameras and computerized light sources that are both programmable. One of the prominent directions is to introduce new features that go beyond classic film-based optical imaging. For instance, the appearances of image parts can be switched on and off using coded exposures and illuminations.

Such special effects were certainly tricky or even impossible to achieve with a standard camera photo-sensor and incandescent light bulb. The post-capture image processing cannot revive the lost effects in general cases. Therefore, the design of infrastructure is normally the biggest challenge in computational photography. Fortunately, electrical engineering and semiconductor fields have invented many kinds of digital hardware, such as structured lights, fluttered shutters, and demodulating sensors. The expansion of photographing capabilities unlatches a new definition of generalized visual effects and countless image applications.

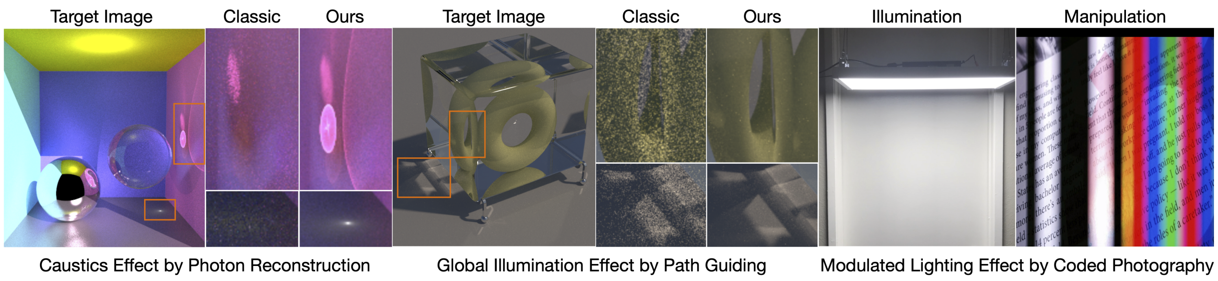

Target Image
Classic
Ours
Target Image
Classic
Ours
Illumination
Manipulation
Caustics Effect by Photon Reconstruction
Global Illumination Effect by Path Guiding
Modulated Lighting Effect by Coded Photography

Figure 1.6. Overview of performance on computing images with different illumination effects including caustics from glass objects, global lighting from single and multiple scattering, and modulated pattern from coded illuminations. Our novel algorithms can create these phenomenons with higher efficiency and variability than classic solutions.

1.4 Overview of Techniques and Contributions

In this section, we show a glimpse of the primary research outcome which will be covered in the rest of chapters, from the algorithm and performance perspectives.

1.4.1 Generating Concentrated Caustics with Glass

The first effect under our consideration is an unusual but vital lighting phenomenon that is regularly known as caustics. It is caused by the light concentration on small regions from transparent or translucent materials, as demonstrated in Figure 1.6 (left). This is readily observable in particular cases, such as swimming pools and glass-like containers, contributing significantly to image photorealism.

However, generating a sharp caustics pattern is not as trivial as it may look, and it has troubled researchers for many years. The specularity nature makes the standard ray tracing algorithm struggle to discover plausible scattering paths between camera and lights. In Chapter 2, we present an alternative image synthesis algorithm that relies on simulating photon particles emitted from the light source. More specifically, we invented a learning-based radiance reconstruction framework tuned for caustics, leading to a sharper and clearer effect on the resulting image compared to classic path tracer as shown in Figure 1.6 (left). Our design can reduce the pain when it comes to rendering scenes like underwater.

1.4.2 Sampling and Reconstructing Global Illumination

Another class of effects originates from high-order scattering events. In notably challenging cases, light sources can be concealed from immediate visibility and illuminate the scene indirectly. In other words, paths of both single and multiple bounces must be constructed side by side to incorporate such global illumination effect.

Unfortunately, the standard path tracing experiences a high failure rate of uncovering locations of lights by randomly sampling the reflected direction at every hit. In accordance with the theory of Monte-Carlo sampling, the chances of building feasible long paths can be tremendously increased using a superior sampling strategy to guide directions of subsequent path segments. In Chapter 3, we strive to maximize the correlation between the sampling distribution and incident illumination by leveraging the concept of online learning. As a result, the pixel variance is greatly reduced on the image, particularly when a series of reflection and refraction events take place in Figure 1.6 (middle). Applying our light transport framework is especially rewarding for tangled scene setups where indirect lighting dominates the image appearance.

1.4.3 Customizing Appearances with Modulated Lighting

Unlocking novel effects from the art of digital photography is the last subject that will be covered in this dissertation. The target appearance is typically driven by the application objective; thus, we build dedicated cameras and illumination modules to evaluate the performance on the captured image. Figure 1.6 (right) presents a special banding effect example that is beyond the capability of ordinary image synthesis.

Under the above circumstance, it is preferable to consider the usage of such unconventional effects prior to the algorithm design. In Chapter 4, we study on a specialized occasion where sensitive contents should be visually protected from wild captures and restricted exclusively to certain cameras. More concretely, we design a digital waveform to create blinking illumination patterns on a customized LED module, and a programmed rolling shutter on authorized cameras to undo the lighting modulation. As demonstrated in 1.6 (right), our computational photography prototype is capable of disturbing the image appearances of prohibited captures by injecting striped effects. Additionally, our enhanced digital encoding supports advanced effects such as image watermarking and visual certification.

1.4.4 Additional Work Done through my Doctoral Career

Apart from the aforementioned research on computing diverse illumination effects, many other image-related projects have also been conducted with my colleagues for the past five years.

From Images to 3D Understanding. Thanks to the development of 3D sensors, the visual data is no longer restricted to images. Modern 3D representations such as depth views, meshes, volumes, and point clouds have started to replace images in certain applications. For this purpose, we have made multiple discoveries on triangle meshes and point cloud scans, which includes a large-scale dataset with comprehensive 3D structures for object-level shape understanding [106], a learning-based 3D reconstruction pipeline from multi-view images [17], and a robust algorithm to process 3D LIDAR point cloud for autonomous vehicles [128]. The transition from 2D images to 3D shapes enhances the capability of regular screen-based photography so that we can reach beyond the binocularity of human vision.

Towards Efficient Image Classification. Perception is another prime task to acquire a high-level semantic understanding of appearances. For instance, assorting images into meaningful categories is a fundamental skill of machines and robotics for perceiving and reasoning. Such classification is non-trivial because of diverse textures and appearances, triggering the prosperity of deep learning and the ImageNet dataset on this subject. To elevate the efficiency of the machine learning approach, we invented a cost-effective binarized neural network for fast image classification using statistical learning [211] and subsequently optimized it on FPGA circuits [40, 39]. Our lightweight design facilitates the visual perception of images captured by low-end mobile devices such as smartphone cameras.

Image-Centric Ubiquitous Computing. The world is surrounded by all kinds of sensors, not only cameras but also photodiodes, radios, and acoustics. These auxiliary sensations are repeatedly connected with images to ease our lives. By introducing proper computational algorithms, we can leverage such ubiquitous sensing signals with images to promote human-computer interactions. By this time, we have activated a few successful innovations comprising an image-based visible light positioning system [209], a structured illumination to produce invisible 2D QR codes [208], and a walking pattern monitor by radars trained from images [104]. Our utilization of images has been proven to be effective in distinctive sensing applications.

Besides the work completed at the university, my explorations over images have fostered a few industry applications throughout internships. During my employment at Apple, we built a high-quality reconstruction pipeline of image dithering for Apple Displays. At Disney, we constructed a 3D pose estimation system using images to enhance guest experiences in theme parks, and a real-time neural rendering algorithm for interactive image synthesis. Weta Digital offered me an opportunity to design a groundbreaking light transport sampling method to manage millions of illuminations, which was pushed into the studio's visual effect pipeline. At Pixar, we pioneered a denoising framework for volume renderings like clouds, smokes, and fires. It fulfills the expected image look from artists and is in the process of being integrated into animation productions. Overall, it is a delightful experience to develop visual computing technology at those world-leading institutions.

Although the above research projects are not the primary focus of this dissertation, they serve as great inspirations by inspecting images from fresh perspectives and interpreting images with new meanings.

1.5 Structure of Chapters

In this dissertation, we separate the discussion of image computations into three parts according to the illuminated effects. In Chapter 2, we first introduce how the special caustics effect is constructed. Next, we progress with computing the generalized global illumination in Chapter 3. Moving forward, we extend the domain of virtual image synthesis to real-world photography in Chapter 4 by demonstrating a customized banding effect on pictures. Finally, we conclude the materials and prospect the future in Chapter 5.

Chapter 2

Creating Vital Lighting Effects on Images

2.1 Challenges of Making Caustics

The *caustics effect* is a special case of global illumination and one of the most challenging phenomenons to create during image synthesis. The underlying principle is the light concentration from the existence of transparent objects such as glass and water. Standard path tracing, however, is particularly inefficient in this situation due to inferior Monte-Carlo samples being used, as illustrated in Figure 2.1.

Tracing particles reversely from light sources is a preferable choice for the case above, resulting in sharper caustics compared to the path tracing approach. Unfortunately, this line of work requires launching a massive amount of photon particles, which cuts back the advantage. Apparently, the space for improvement is still quite substantial for making beautiful caustics images. In this chapter, a faster approach is presented to reconstruct those crucial caustics using a lot fewer particles, which has been published in [212].

2.2 Overview

Recently, deep learning-based denoising approaches have led to dramatic improvements in low sample-count Monte Carlo rendering. These approaches are aimed at path tracing, which is not ideal for simulating challenging light transport effects like caustics, where photon mapping is the method of choice. However, photon mapping requires vast numbers of traced photons

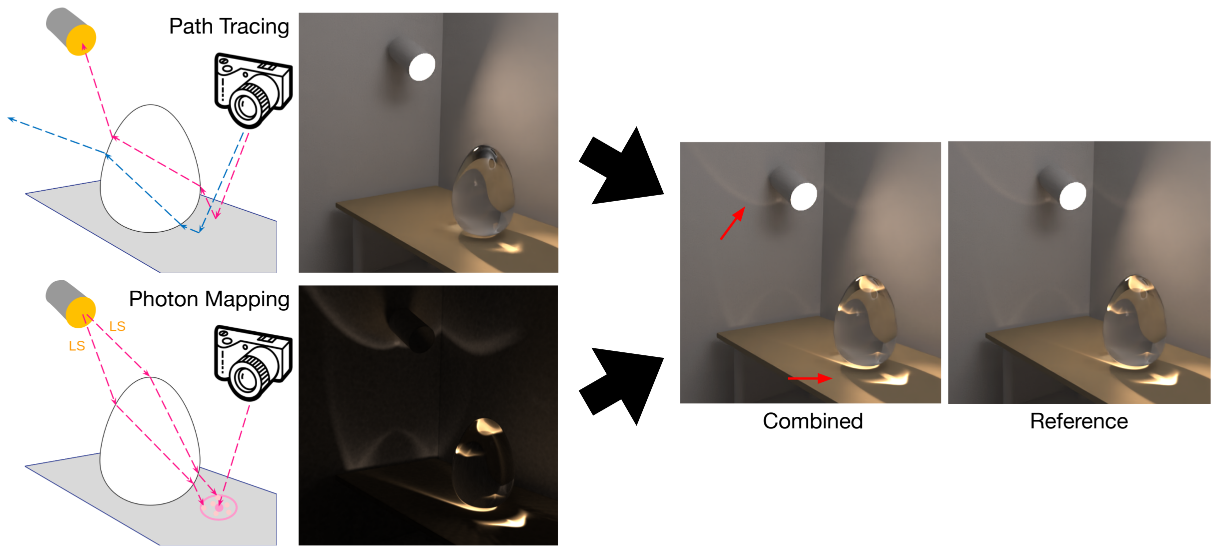


Figure 2.1. Challenges of generating path-traced caustics. On the contrary, photon mapping performs fairly well by reconstructing such effects from light specular (LS) particles traveling through the transparent surfaces. By combining the output of both methods, we can produce images with expected reference looks.

to achieve high-quality reconstructions. In this work, we develop the first deep learning-based method for particle-based rendering, and specifically focus on photon density estimation, the core of all particle-based methods. We train a novel deep neural network to predict a kernel function to aggregate photon contributions at shading points. Our network encodes individual photons into per-photon features, aggregates them in the neighborhood of a shading point to construct a photon local context vector, and infers a kernel function from the per-photon and photon local context features. This network is easy to incorporate in many previous photon mapping methods (by simply swapping the kernel density estimator). It can produce high-quality reconstructions of complex global illumination effects like caustics with an order of magnitude fewer photons compared to previous photon mapping methods. Our approach essentially reduces the required number of photons, significantly advancing the computational efficiency in photon mapping.

2.3 Introduction and Methodology

Computing global illumination is crucial for photorealistic image synthesis. Ray tracing-based methods have been widely used to simulate complex light transport effects with global illumination in films, animations, video games, and other industrial fields. The most successful approaches are based on either Monte Carlo (MC) integration, like path tracing [81, 169], or particle density estimation, like photon mapping [77]. Photon mapping techniques can efficiently simulate caustics and other challenging light transport effects, which are very hard and even impossible for pure Monte Carlo-based methods to simulate.

In general, both MC-based and particle-based methods require numerous samples to render noise-free images, and are thus computationally expensive. Recently, significant progress has been made in denoising MC images rendered with low sample counts using deep learning techniques [15, 5]. However, there is relatively little work in particle-based methods for low-sample reconstruction and current photon mapping techniques still require a massive number of traced photons to achieve accurate, artifact-free radiance estimation.

We present the first deep learning-based approach for particle-based rendering that enables efficient, high-quality global illumination with a small number of photons. Our approach is particularly good at reconstructing diffuse-specular interactions like caustics, for which previous photon mapping methods require large photon sample counts (and path-tracing at reasonable sample counts can miss altogether). We focus on photon density estimation—a key component of all particle-based methods—and introduce a novel deep neural network that can estimate accurate photon density at any surface point in a scene given only sparsely distributed photons.

Previously, the most successful density estimation methods for photon mapping were kernel-based methods that use traditional kernel functions (like a uniform or cone kernel) to compute output radiance at a surface point as a weighted sum of nearby photons. While previous methods have improved the kernels by controlling the kernel bandwidths or shapes [85, 142, 84], traditional kernel functions still require a large enough count of photons located in a small enough bandwidth around every surface shading point, for which a vast number of photons need to be traced, to compute accurate photon density. In contrast, we propose learning to predict a kernel function at each shading point to aggregate nearby photon contributions effectively. Our predicted kernels leverage data priors and are able to compute accurate photon density estimation for complex global illumination from photon counts that are an order of magnitude fewer than traditional methods.

Our network considers local photons around a queried surface point within a predefined bandwidth as input. Unlike traditional methods that often treat photons individually or leverage standard statistics to aggregate photons, we leverage learned local photon statistics—encoded as a deep photon context vector inferred by the network—around a surface point for per photon kernel weight estimates. Specifically, the network first processes individual photons to extract per-photon features and aggregates them across photons using pooling operations to obtain a deep photon context feature that represents the local photon statistics. The network processes the individual per-photon features concatenated with the local context to compute per-photon kernel weights, which are used to perform density estimation by a weighted sum. We demonstrate that this approach of learning kernel prediction is more efficient than a baseline that directly estimates photon density from the aggregated deep context vector.

To train our network, we create diverse photon distributions by tracing photons in 500 procedurally generated scenes with complex shapes and materials. We sample surface points on diffuse surfaces, which form a 512×512 image (one pixel per point) in each scene, and we compute the ground truth photon density of each point using progressive photon mapping [61] with billions of photons. Note that our network focuses on the local photon distribution properties of surface points. Hence, every surface point in a scene is a training datum, allowing us to train a generalizable network without many images.

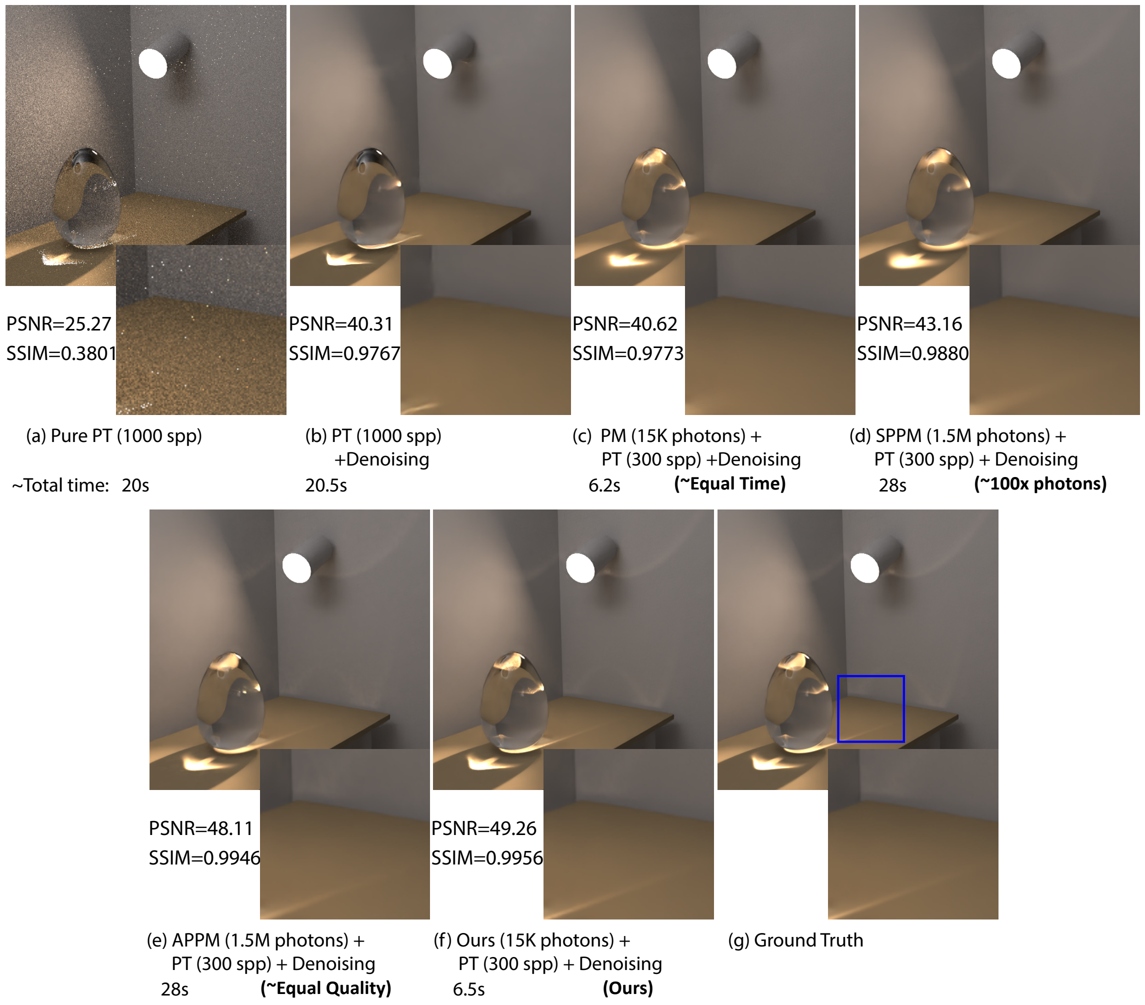


Figure 2.2. We present a novel learning-based photon mapping (PM) method that can be used to synthesize photorealistic images (f) with detailed caustics (shown and compared in the insets) from very sparse photons for scenes with complex diffuse-specular interactions. In particular, we use our method with only 15k photons ($\sim$0.06 photons per pixel) to compute accurate global illumination for light-specular paths. We use path tracing (PT) with a moderate number (300) of samples per pixel (spp) to compute the other paths and apply the Optix learning-based denoiser (based on [15]) to remove the Monte Carlo (MC) noise. In contrast, pure PT leads to noisy results lacking focused caustics (a) even with 1000 spp that is significantly more than our photon and path samples. While this noise can be mitigated using a learning-based denoiser, this introduces artifacts and cannot recover the caustics (b). Combining PT and standard PM [77] with 15k photons, and then denoising (c), avoids these artifacts but still does not reconstruct caustics accurately from such low photon counts. While providing 1.5M photons (this is 100 times the number of photons our method uses) and applying the advanced stochastic progressive PM (SPPM) [62] enables a more accurate result (d), it is still slightly worse than ours. In contrast, our result (f) accurately reproduces the caustic effects in the global illumination, as compared to the ground truth (g), with significantly fewer samples. Ours is comparable with (if not better than) the result from adaptive progressive PM (APPM) [85] with 100 times the number of photons (e).

2.4 Advantages over Traditional Techniques

In Figure 2.2, we demonstrate that, using only 15k photons, our method can synthesize high-quality images. Conversely, path tracing and photon mapping variations fail to do so; even when combined with advanced progressive and adaptive techniques, SPPM and APPM require significantly more samples (1.5M photons) to achieve comparable results. This makes our approach an important step towards making photon mapping computationally efficient. Moreover, our experiments leverage an effective practical hybrid approach: using our method for reconstructing light-specular (LS) paths – the light transport paths that interact with specular surfaces before arriving at light sources—and low sample-count path tracing with learning-based denoising for all other light transport paths. This leverages the advantages of both MC denoising and our efficient photon density estimation technique.

2.5 Related Work

Monte Carlo path integration. Kajiya [81] introduced the rendering equation and Monte Carlo (MC) path tracing. Since then, various methods for MC path integration have been developed, including light tracing [30], bidirectional path tracing (BDPT) [93, 171], and Metropolis light transport (MLT) [169, 123, 21]. These methods can simulate complex light transport with accurate global illumination in an unbiased way. However, pure MC based methods typically require a huge number of samples (traced paths), especially for very low probability paths like the classical caustic or specular-diffuse-specular (SDS) paths. We base our method on the photon mapping technique, which is efficient for caustics and SDS, and we aim to achieve sparse reconstruction.

Monte Carlo denoising. While there is little progress in sparse reconstruction with low sample counts in photon mapping, many approaches have been proposed to achieve MC rendering with low sample counts. A recent survey of sparse sampling and reconstruction is presented by Zwicker et al. [216]. MC denoising methods can be categorized into a-priori methods that rely on prior theoretical knowledge [29, 31, 197, 191], and a-posteriori methods that filter out the noise in rendered images with few assumptions about the image signal [121, 137, 83].

Recently, deep learning techniques have been introduced to achieve MC denoising [15, 5], and many methods utilize kernel prediction [5, 172, 193]. Kalantari et al. [83] propose to predict the parameters of fixed filtering functions using fully-connected neural networks. Bako et al. [5] leverage deep convolution neural networks to predict kernels to linearly combine the original noisy radiances of neighboring pixels. Gharbi et al. [47] make use of individual screen-space path samples and predict a kernel for each sample that splats the radiance contributions to its neighboring pixels. Deep learning techniques have also been extended to gradient-domain rendering [87]. In contrast, we apply deep learning in photon density estimation and leverage local photon statistics for density estimation from sparse photons. Our network considers individual scene-space photon samples around each shading point and predicts a kernel to gather per-photon contributions. Our approach is the first that introduces deep learning in photon mapping and demonstrates learning-based kernel prediction in this context.

Photon density estimation. The rendering equation [81, 73] can be approximated by particle density estimation [145, 77, 177]. Most particle-based methods are based on the original photon mapping framework [77]; it first traces rays from light sources to distribute photons in a scene, and then gathers neighboring photons at individual shading point to approximate radiance estimates. Photon mapping achieves low variance in the rendered images and leads to blurred, less noticeable artifacts at the cost of introducing bias in the estimates. Photon mapping is able to consistently converge to the correct solution by increasing the number of photons towards infinity and reducing the bandwidth towards zero.

Previous work has investigated progressive methods to overcome the memory bottleneck and enable arbitrarily large photon numbers [61, 59, 62, 89], bidirectional methods to improve rendering glossy objects [173], adaptive methods to optimize photon tracing [60], and the combination of unbiased MC methods and photon mapping [45, 63, 46]. Many relevant works have been presented to improve the kernel density estimation by utilizing standard statistics for adaptive kernel bandwidth [78, 85, 84] or anisotropic kernel shapes [142]. Other works leverage ray differentials [141], blue noise distribution [152, 153, 154], traditional linear regression [70] and Gaussian mixtures fitting [75] to improve the reconstruction. In contrast, we focus on accurately computing photon density with sparse photons, which has not been explored in previous work. Essentially, we replace the traditional kernel density estimation with a novel deep learning-based module, and keep the rest unchanged in the standard photon mapping framework. This potentially enables the combination of our technique and previous photon mapping techniques that focus on other components in the framework.

2.6 Fundamentals of Particle-Based Photon Mapping

2.6.1 Radiance Reconstruction Formulation

Photon mapping techniques compute reflected radiance via density estimation. Kernel density estimation [178] is the most widely used density estimation method in statistics, and has been widely applied in photon mapping. Early works use the uniform kernel that treats nearby photons equally [77, 61]; subsequent works extend photon density estimation to support arbitrary smooth kernels [62, 89]. In general, the reflected radiance at a shading location $\boldsymbol{x}$ is computed by:

$$L(\boldsymbol{x}, \boldsymbol{\omega}) \approx \frac{1}{N} \sum_{i=1}^{N} k_r(\boldsymbol{x}, \boldsymbol{x}_i) \tau_i, \tag{2.1}$$

where N is the total number of photon paths that are emitted in a scene, $\boldsymbol{\omega}$ is the reflected direction, $\boldsymbol{x}_i$ is the location of a photon, τ_i is the photon contribution and k_r represents the kernel function with bandwidth r. In general, the photon contribution τ_i is the product of the BRDF and the photon energy. In this work, we only compute photon density on diffuse surfaces, as is done in many classical photon mapping methods. In this case, the BRDF at a shading point is ρ/π, where ρ is the albedo. Correspondingly, $\tau_i = \phi \rho / \pi$, where ϕ represents the accumulated path contribution divided by the sampling probability, which can also be interpreted as the energy flux carried by the photon. Therefore, $\boldsymbol{\omega}$ can be removed and ρ can be taken out of the summation in Equation 2.1. We therefore consider the photon energy ϕ as the photon contribution in this work.

2.6.2 Density Estimation Kernels

The kernel k_r assigns linear weights to photons, which are used to linearly combine the contributions of photons in a local window with radius r. Traditionally, k_r is a uniform function $(1/(\pi r^2))$ or a function of the distance from the shading point to a photon $(\|\boldsymbol{x}-\boldsymbol{x}_i\|)$. Instead, we propose to leverage data priors to predict kernels to aggregate photon contributions.

2.7 Learning to Estimate Photon Density

In this section, we present our learning-based approach for photon density estimation. Our approach is lightweight and focuses on density estimation only; we keep the main framework of standard photon mapping and upgrade the traditional, distance-based and photon-independent kernel functions (k_r in Equation 2.1) to novel, learned and local-context-aware kernel functions represented by a deep neural network (see Figure 2.3).

In particular, given a shading point, our network considers its K nearest neighbor photons, which adaptively selects the bandwidth r. Multiple properties of individual photons are used as input for the network, including photon positions $\{\boldsymbol{x}_i\}_{i=1}^K$, photon directions $\{\boldsymbol{d}_i\}_{i=1}^K$ and photon contributions $\{\phi_i\}_{i=1}^K$. We also supply the number of nearest photons K to the network to let it better understand the local photon distribution. Our network (denoted as Φ) regresses per-photon kernel weights to compute radiance estimates via a weighted sum similar to Equation 2.1:

$$L(\boldsymbol{x}) \approx \frac{\rho}{N\pi r^2}\sum_{i=1}^{K}\Phi_{r,i}(\boldsymbol{x},\{\boldsymbol{x}_i\},\{\boldsymbol{d}_i\},\{\phi_i\})\phi_i, \qquad (2.2)$$

where $\Phi_{r,i}$ represents the predicted kernel weight for photon i. Note that, our network uses information about *all* photons in a local neighborhood for per-photon kernel prediction; it obtains deep photon statistics and associates per-photon information with statistical context to compute kernels for photon aggregation.

2.7.1 Processing Photon Point Cloud

Photon distributions are highly diverse across shading points and scenes, making it challenging to design a network that generalizes to different inputs. Besides, deep neural networks are known to benefit from normalized input data to correlate values from different domains. Therefore, we pre-process the input photon properties to allow for better generalizability and performance.

Since light intensities can have a very high dynamic range (HDR), the photon contributions τ_i can vary widely in range, which is highly challenging for a network to process. We introduce a mapping function to pre-process the photon contributions,

$$t_a(u) = \frac{\log(u+a) - \log(a)}{\log(u+a) - \log(a) + 1}, \tag{2.3}$$

where $a = 0.01$ is an additional parameter. Essentially, $t_a(u)$ maps HDR values u from $[0, \infty]$ to $[0, 1]$. We further linearly map these values to $[-1, 1]$ and provide them as network input. We observe that such a mapping process facilitates the network learning.

For photon positions $\boldsymbol{x}_i$ and directions $\boldsymbol{d}_i$, we first transform them into the local coordinate frame of the shading point; the coordinate frame is constructed using the position and normal of the shading point and two orthogonal directions that are randomly selected in the tangent plane. This transforms the network inputs into a consistent coordinate system and improves generalizability.

The bandwidth r of our learned kernel is determined by the distance of the K^{th} nearest photon. This leads to an extensive range of bandwidth values given various photon distributions, which is highly challenging for a deep neural network to process. Motivated by the bandwidth normalization used in traditional kernels [178, 145], we divide the photon positions in the local coordinates by the bandwidth r, and scale the final density estimates by $1/r^2$, which is shown

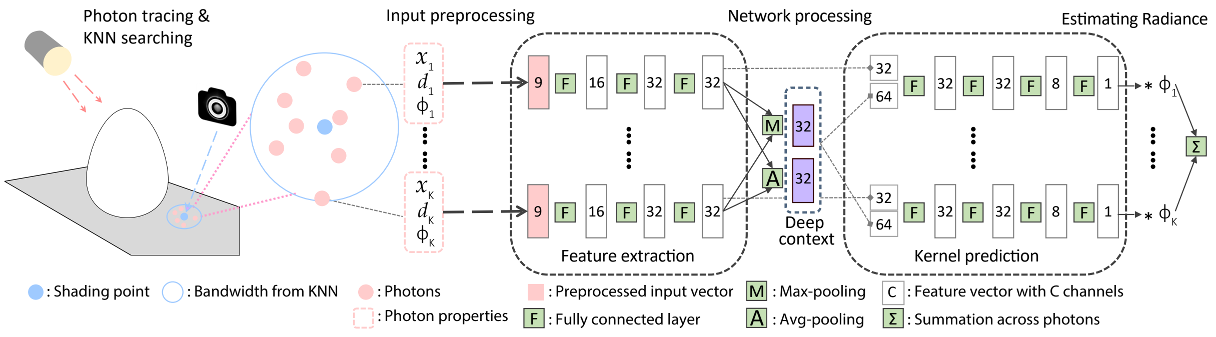


Figure 2.3. Overview of our deep photon density estimation network. Given a set of photons within the bandwidth of a shading point, we pre-process these photons' properties and input them to feature extractor MLPs that compute per-photon features. These are aggregated using max- and average-pooling to construct a deep context feature. The original per-photon features and the deep context are concatenated and processed by a kernel prediction MLP that predicts a kernel weight. Finally, these kernel weights are used to sum the photon contributions and produce the reflected radiance.

in Equation 2.2. This normalizes all input photon positions into a unit sphere and post-scales the computed photon density by the actual window area. As a result, our network is invariant to the actual bandwidths, and effectively generalizes to different photon distributions and supports different numbers of *total emitted photons* that will introduce different bandwidths for the same K.

Note that, different terms of our network input are all normalized into the range of $[-1, 1]$, which enables our network to correlate and leverage different photon properties from various domains in an efficient way. Our input pre-processing also makes our network translation-, rotation-, and scale-invariant to diverse photon distributions, leading to good generalization across different scenes and different numbers of emitted photons.

2.7.2 Kernel-Predicting Reconstruction Architecture

The inputs to our network are essentially a set of multi-feature 3D points in a unit sphere. There is no meaningful inherent point ordering in the set, and the number of points (K) is not fixed. We thus leverage PointNet [127] style neural networks with multi-layer perceptrons, which accept an arbitrary number of inputs and are invariant to permutations of inputs. As shown in

Figure 2.3, our network consists of two sub-networks, a feature extractor and a kernel predictor; they are both fully connected neural networks and process each photon individually.

The feature extractor first processes each individual photon; it considers the pre-processed photon properties (9 channels including positions, directions and contributions) as input, and extracts meaningful features using multi-layer perceptrons. Specifically, we use three fully connected layers in the feature extractor, and each layer is followed by a ReLu activation layer. The feature extractor leverages linear and non-linear operations to transform the original input into a learned 32-channel feature vector. These per-photon features are then aggregated across photons by max-pooling and average-pooling operations which output the deep photon context vector. This vector represents the local photon statistics in a learned non-linearly transformed space. The kernel predictor then leverages the across-photon context and the per-photon features to predict a single scalar that represents the kernel weight for each photon. These per-photon kernel weights are the final output of our network and will be used to linearly combine the original photon contributions as expressed in Equation 2.2. The kernel predictor is also a three-layer fully connected neural network with ReLU as activation layers, which is similar to the feature extractor but with different channels at each layer.

Note that, unlike previous work that treats each photon independently, we propose to correlate per-photon information with local context information across photons. Our feature extractor transforms photon properties into learned feature vectors, which allows for collecting photon statistics in the learned neural feature space to obtain the photon context for the following kernel prediction. Our whole network is very lightweight and involves only six fully connected layers; this ensures a highly efficient inference process. We show that such a lightweight network is able to reconstruct accurate photon density from sparse photons effectively.

2.7.3 Dataset and Training Details

Data generation. Monte Carlo denoising usually requires a large number of images to train and is hard to generalize across different types of scenes. Our method focuses on local

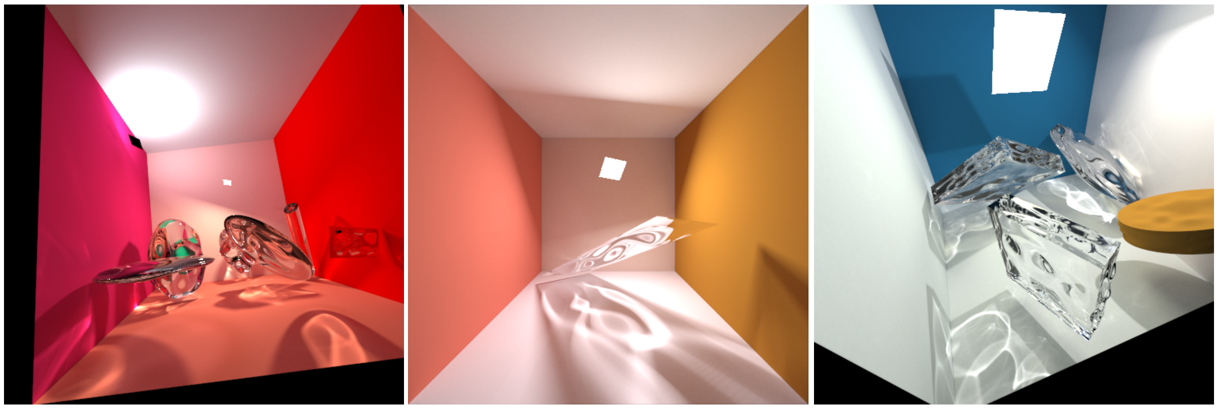


Figure 2.4. Examples of our procedurally generated training scenes.

photon distributions; in other words, to learn proper data priors, we desire the diversity of photon distributions in terms of individual shading points and not necessarily of the entire scenes. This allows for good generalizability of our network with a relatively small number of training scenes, which can even be very different from our final testing scenes. Inspired by [195, 196], we procedurally create shapes from primitives with random sizes and random bump maps; a set (randomly from 1 to 16) of such shapes is then placed in a box and distributed roughly as a grid. We also place multiple area lights with random locations and rotations in the scene, and randomly assign specular materials and diffuse materials to the scene objects.

A few examples of these scenes are shown in Figure 2.4; complex light transport effects with diverse photon distributions are simulated. To sample shading points in each scene, we shoot rays from a camera through an image plane with 512×512 pixels and select the first diffuse intersections as target shading points. We trace photons from light sources and keep the ones that contribute to indirect lighting. Progressive photon mapping [61] is then applied to compute ground truth photon densities for each point with a total number of about 1 billion photon paths. For each scene, we store 10 million photon paths and a 512×512 multi-channel image containing ground truth radiances and other necessary information (positions, normals, and BRDFs) of shading points. We create 500 scenes for training our neural networks and test our network on scenes that are significantly different from our training data (see Figure 2.2 and

Figure 2.9).

Loss function. We supervise our network with the ground truth radiance estimates. The final radiances are in high dynamic range, which can easily make the training dominated by high-intensity values; we therefore tone-map the radiance estimates using the μ-law as in [82]. The mapping function $p_\mu(v)$ is given by:

$$p_\mu(v) = \frac{\log(1+\mu v)}{\log(1+\mu)}, \tag{2.4}$$

and we set $\mu = 5000$ following [82]. We tone-map both our estimated radiance and the ground truth radiance, and we apply L_2 loss on the mapped values.

Training parameters. We randomly select K from 100 to 800 and use from 0.3 million to 4 million photons to train our network, which makes it generalize well to various bandwidths and photon counts. We use Adam to train our network for 6000 epochs with an initial learning rate of 10^{-4} and a batch size of 2000 random shading points.

2.8 Implementation and Evaluation

We now present a comprehensive evaluation of our method. A variety of scenes with complex caustics effects are selected for experimentation.

2.8.1 Verification Study

We first justify the choices of our network design. In particular, we compare our network with a baseline network that estimates the final radiance without predicting kernels; this comparison network has a similar architecture but directly outputs the final irradiance from the across-photon deep context vector. Figure 2.5 shows the training processes of these networks; our network converges significantly faster than the baseline method. This demonstrates the effectiveness of combining kernel density estimation and deep learning and is consistent with previous results on denoising for path tracing [5, 172, 47].

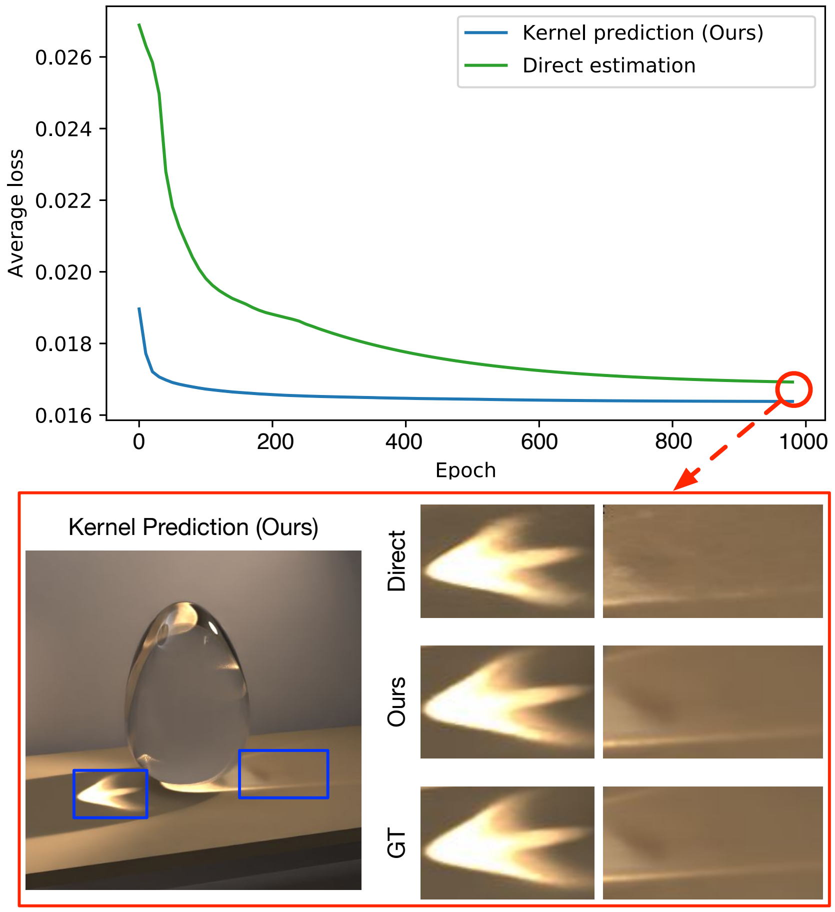


Figure 2.5. Kernel prediction. We compare the optimization speed of our kernel-prediction network and a baseline direct-estimation network. Our network converges faster to the lower loss value. On the bottom, we show the rendered images from the two networks trained with 1000 epochs. Our results are closer to the ground truth with more details in the caustics.

2.8.2 Evaluation Scenes and Photon Generation

We evaluate our method on six challenging scenes (EGG, WINE, RINGS, POOL1, POOL2, DRAGON) that involve complex caustics and other diffuse-specular interactions with LS paths. In theory, path tracing can never reconstruct LS paths if we use a point light source; we therefore use area lights in the scenes to allow for reasonable comparisons with PT. For each scene, we shoot photons for 0.1 seconds, which generates about 0.8M photon paths with a maximum of five photons per path; we only keep those photons that involve light-specular paths in the scenes. We denote the number of valid photons we consider as M, which is a number that is different from the total emitted photon paths N in Equation 2.1. Because of various compositions of scenes, there are 15k (EGG), 85k (WINE), 77k (RINGS), 50k (POOL1), 100k (POOL1) and 125k (POOL1) valid photons that are used in the six scenes respectively. We also evaluate with the number of photons that are traced in one second—corresponding to ten times the number of photons traced in 0.1 seconds—to justify the generalization of our network to different numbers of emitted photons, and compare with the other methods with photons that are traced in ten seconds to justify the quality of our sparse reconstruction.

2.8.3 Combining Denoising and Deep Photon Mapping

We evaluate our deep photon density estimation by combining our method with MC denoising. Specifically, we apply our learning-based density estimation only to compute the challenging light transport effects which involve LS paths that are extremely hard to trace in PT and likely to introduce caustics. In addition, we use path tracing with relatively low sample counts to compute the remaining light transport paths, and use modern learning-based denoising—the Optix built-in denoiser based on [15]—to remove the MC noise.

By removing LS paths in PT, we also make PT and MC denoising much easier. As shown in Figure 2.6, PT without LS paths can be effectively denoised using modern learning-based denoising techniques with 100 spp, whereas full PT with LS paths introduces extensive noise with the same 100 spp, causing denoising to fail completely. In fact, the standard PT plus denoising pipeline is not able to recover the complex light transport effects with even 1000 spp (see Figs. 2.2, 2.7). In contrast, we demonstrate a practical way of combining our efficient deep photon mapping with MC denoising for photorealistic image synthesis, in which we leverage the benefits of low-sample reconstruction in both scene-space particle density estimation and screen-space MC integration.

Table 2.1. Quantitative RMSE evaluation. We test our networks trained with different K ($K = 50$ and 500, denoted with Ours-K) on six novel scenes with different numbers of valid photons (M). We also test a variant of our network architecture with enlarged four times capacity (Ours-Large) using the same K. We compare RMSE against standard photon mapping (PM) [77] under the same conditions, and also progressive PM (PPM) [61] and adaptive PPM (APPM) [85]. We highlight the best and the second-best results in red and blue for each row; note that, all of them are our results. We also highlight the best result of the comparison methods in yellow, which is often worse than any of our network settings.

Scene	(M)	Ours-50	Ours-L-50	PM-50	Ours-500	Ours-L-500	PM-500	PPM	APPM
EGG	(15k)	0.013	0.006	0.085	0.013	0.006	0.165	0.085	0.080
	(150k)	0.012	0.006	0.036	0.008	0.004	0.079	0.065	0.043
	(1.5M)	0.013	0.007	0.031	0.006	0.003	0.027	0.030	0.030
WINE	(85k)	0.052	0.028	0.116	0.044	0.021	0.222	0.134	0.111
	(850k)	0.035	0.027	0.053	0.023	0.014	0.102	0.064	0.047
	(8.5M)	0.032	0.030	0.045	0.014	0.011	0.037	0.031	0.026
RINGS	(77k)	0.042	0.023	0.069	0.023	0.008	0.153	0.137	0.143
	(770k)	0.041	0.024	0.046	0.011	0.006	0.042	0.050	0.049
	(7.7M)	0.045	0.020	0.066	0.012	0.009	0.023	0.017	0.014
POOL1	(50k)	0.244	0.174	0.281	0.214	0.146	0.323	0.327	0.277
	(500k)	0.214	0.173	0.221	0.135	0.115	0.244	0.249	0.193
	(5.0M)	0.237	0.186	0.259	0.107	0.105	0.124	0.206	0.125
POOL2	(102k)	0.178	0.125	0.226	0.167	0.095	0.260	0.262	0.224
	(1.0M)	0.132	0.121	0.147	0.115	0.080	0.221	0.211	0.155
	(10.2M)	0.134	0.128	0.159	0.066	0.061	0.102	0.163	0.088
DRAGON	(125k)	0.066	0.054	0.073	0.052	0.043	0.089	0.126	0.102
	(1.2M)	0.056	0.054	0.061	0.034	0.033	0.044	0.083	0.054
	(12.5M)	0.059	0.059	0.078	0.028	0.027	0.031	0.059	0.035

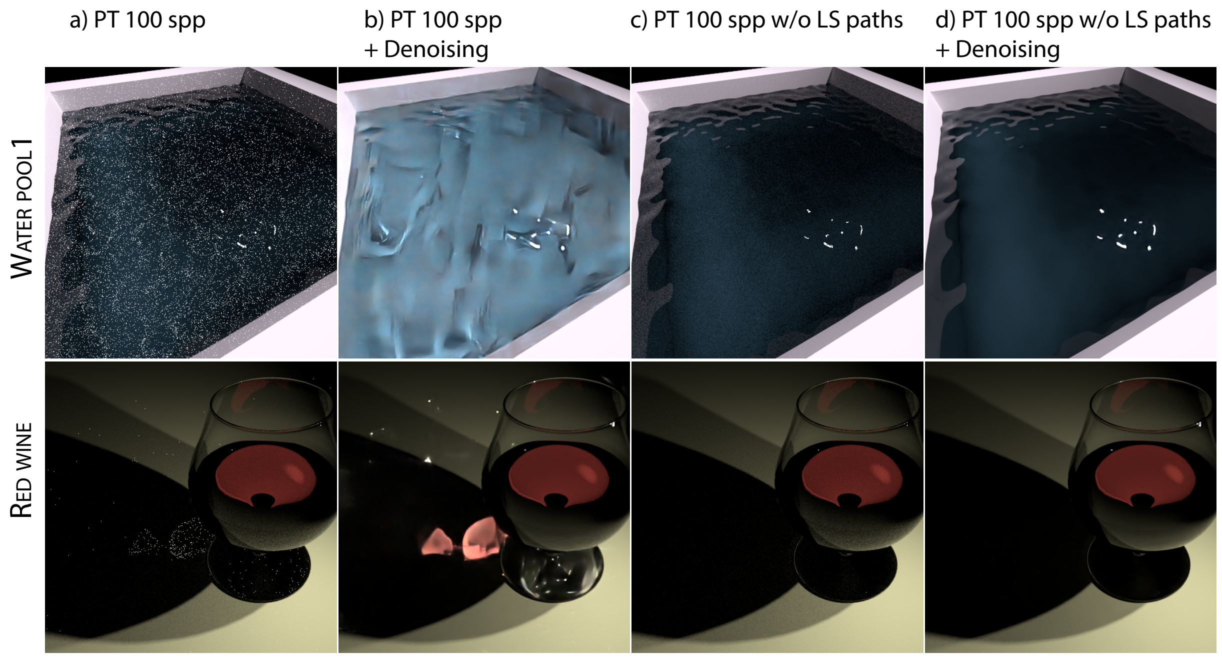

a) PT 100 spp
b) PT 100 spp + Denoising
c) PT 100 spp w/o LS paths
d) PT 100 spp w/o LS paths + Denoising
WATER POOL 1
RED WINE

Figure 2.6. Path tracing (PT) with and without light-specular paths (LS). We show PT and denoising results using 100 spp with and without light-specular paths. The noise can be seen more clearly when zooming into the electronic PDF.

2.8.4 Investigation on Photon Parameters

We observe that it is tough for a single network to generalize across different numbers of input photons (K). We thus use a fixed K when training per network, and specifically we train two networks with $K = 50$ and $K = 500$ for the evaluation. We also compare with a variant of our network that has four times the channels at each layer in our network architecture to evaluate if larger network capacity leads to higher performance. This large network generally leads to better performance (see Table 2.1), but it requires about three times longer inference time (see Table 2.2); please see the following parts in the section for more discussion about quality and performance. In the experiments, we use DPM (deep photon mapping) to denote the network with regular capacity and DPM-L (or Ours-L) to denote the one with a larger capacity.

In all experiments, we compare with the classical photon mapping (PM) with the identical k-NN photons as inputs. We also compare various methods that are designed to reduce the bandwidth with large photon counts progressively. In particular, for density estimation at fixed

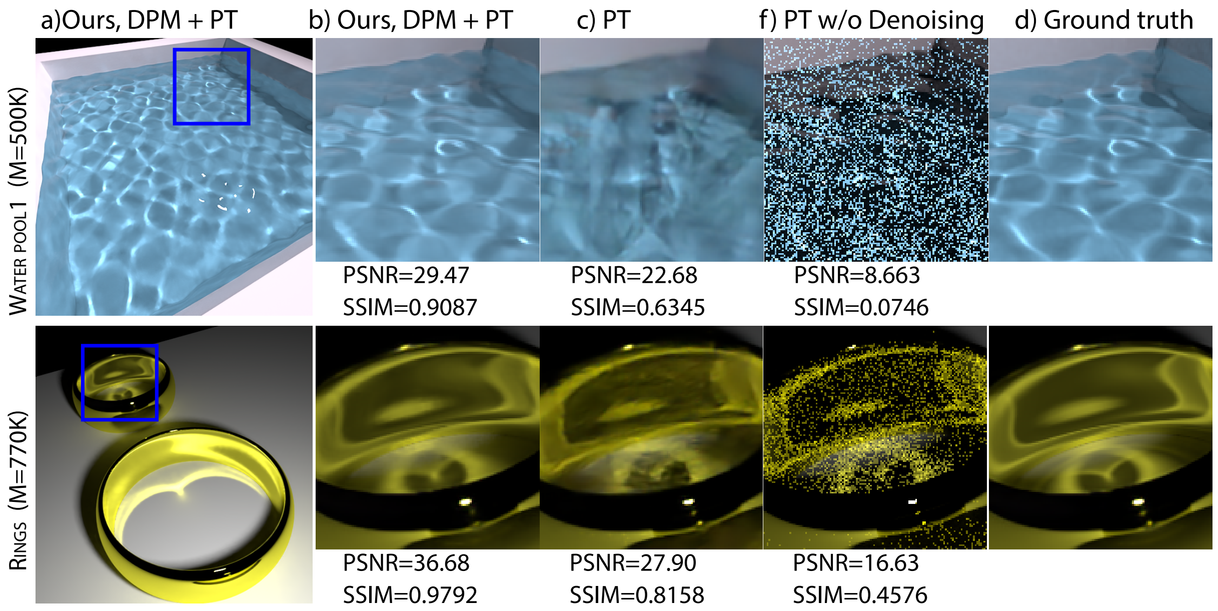


Figure 2.7. We show our final results in full images (a). Our final results are computed by combining our deep photon mapping results and path tracing with denoising. We compare against pure path tracing using 1000 spp with (c) and without (d) denoising on insets. Obviously, path tracing alone even with 1000 spp cannot handle the LS paths.

surface points, we compare with progressive photon mapping (PPM) [61]. Given a certain number of input photons, the quality of PPM is influenced by the initial radius and the number of photons per iteration. To make a fair comparison, we compare 30 different variants (10 radii and 3 photon counts per iteration) of the two parameters and choose the best settings (with lowest RMSEs) for each scene. We also compare with adaptive progressive photon mapping [85] similarly using the best radius and number of photons per iteration from 30 different variants of parameters. For visual comparisons, we compare with stochastic PPM (SPPM) [59], when there are transparent surfaces in a scene that require sampling multiple surface points per pixel.

2.8.5 Quantitative and Qualitative Results.

We now evaluate our method quantitatively and qualitatively with different numbers of photons counts (M) and different variations of training parameters (input photon number K and capacity). Table 2.1 shows quantitative RMSE evaluation of photon density estimation on the six testing scenes; the numbers are averaged across about 260k surface shading points sampled

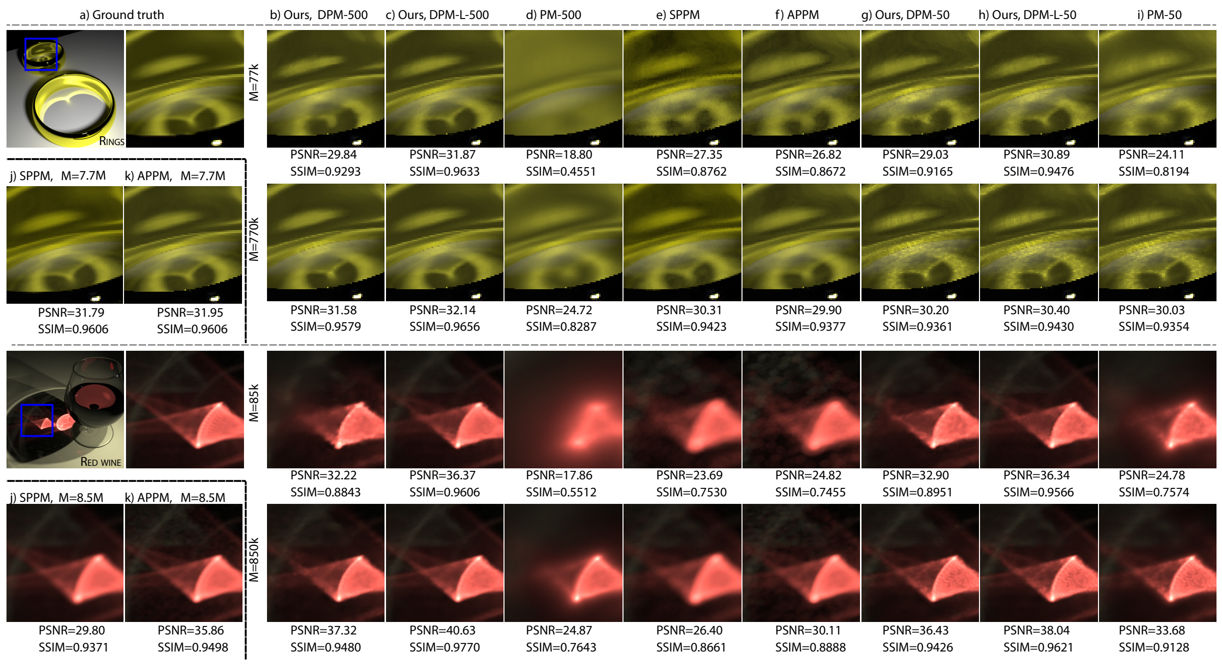


Figure 2.8. We show results of our method with different numbers of input photons (K). We compare against PM, SPPM, and APPM with the same number of total photons (M) on insets marked in the left-top ground truth image. We also show the results of APPM and PPM with ten times the largest number of photons our method uses. The PSNRs and SSIMs of the insets are shown correspondingly.

by tracing rays from a camera and selecting the first diffuse hit points in the scenes. Note that, across all these different scenes with different photon counts, our method with $K = 500$ performs consistently better than all the comparison PM methods, including standard PM [77], PPM [61] and APPM [85], with the same number of total photons. Most of our results are better than PM and PPM with ten times the photon count as ours. APPM leverages traditional statistical information of local photons to improve the density estimation of PPM, which can achieve fairly good results; however, it requires the number of photons to be large enough to obtain good statistics. In contrast, our method leverages learned statistics in the network, which achieves significantly better results than APPM with the same number of photons; ours is actually comparable to the APPM that uses ten times the total number of photons. Note that, the APPM and PPM results are selected from tens of APPM and PPM variants with different hyper-parameters for their best performance; yet, our method still outperforms the best of these

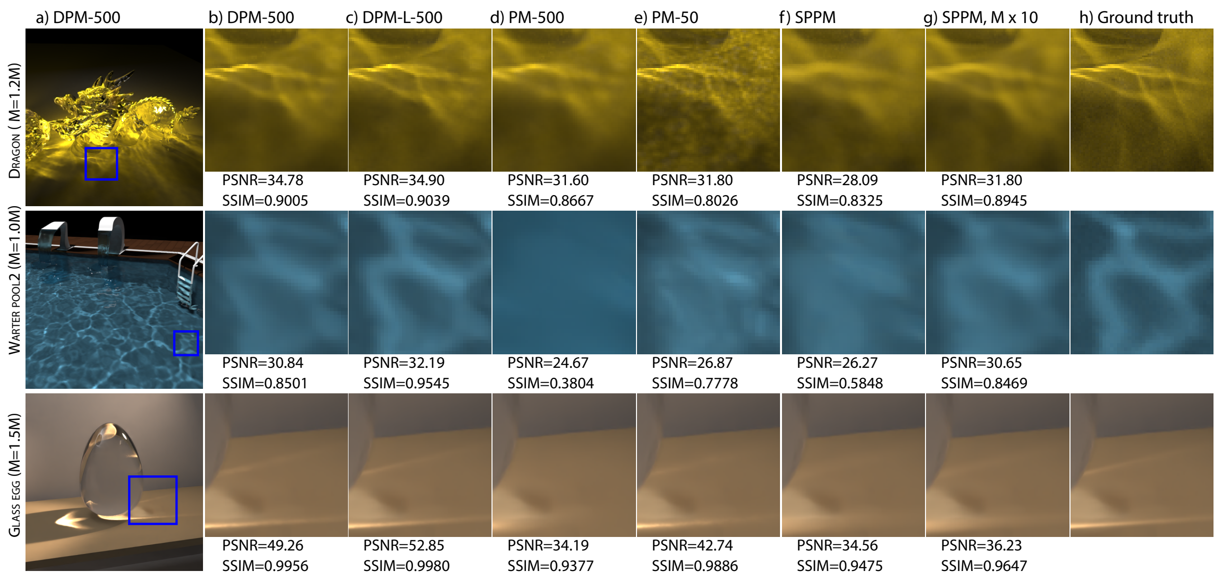


Figure 2.9. We show our results on full images (a). We compare against PM with the same input photons (d) and SPPM with the same (f) and ten times (g) the total photon counts on insets. PSNRs and SSIMs are also calculated for all insets and listed below.

variants.

To visually illustrate the numbers in Table 2.1, we demonstrate all the rendering results of RINGS and WINE with the first two rows (first two M) in Figure 2.8; we also show the visual results of APPM and PPM with larger M in Figure 2.8(j)(k). Additionally, we show the results of three testing scenes in Figure 2.9, where we compare our DPM-500 with PM and SPPM. In Figure 2.2, we show the result of our DPM-50 and compare it with PT, PM, SPPM, and APPM. In general, our method with $K = 500$ outperforms the comparison methods with the same number of photons qualitatively and quantitatively. Moreover, our results are comparable to (if not better than) the comparison methods that use ten times the number of photons in the scene. While the larger network with $K = 500$ (Ours-L-500) performs better than the regular network, the larger one also requires a longer inference time (see Table 2.2). Therefore, our regular network with $K = 500$ is generally the best choice for most cases, which stably achieves high-quality results. However, when the timing is not a critical issue, the large network will be a better choice for higher accuracy.

In most cases, the Ours-500 ($K = 500$) results are better than the Ours-50 ($K = 50$)

Table 2.2. Timing. We show the corresponding running time in seconds for each photon mapping component. Our experiments are run with photons that are traced within 0.1s, 1.0s, and 10.0s in each scene. We list the corresponding gathering time to find the neighboring photons for about 512×512 surface shading points. The numbers of total photons are also shown, corresponding to the M in Table 2.1. We list the network inference time for 512×512 surface shading points for our regular network (DPM) and a large network (DPM-L) with $K = 50$ and 500. Note that, the network inference time is determined by its capacity and K, and is independent of the number of total photons in the scene.

Tracing	Gathering	#Photons	DPM-50	DPM-L-50	DPM-500	DPM-L-500
0.1s	0.12s $\sim$ 0.5s	15k $\sim$ 125k	0.3s	1.0s	3.0s	10.0s
1.0s	1.2s $\sim$ 5.0s	150k $\sim$ 1.2M	0.3s	1.0s	3.0s	10.0s
10.0s	12s $\sim$ 50s	1.5M $\sim$ 12M	0.3s	1.0s	3.0s	10.0s

ones, indicating that our network is usually in favor of more nearest neighbor photons (K) as input. Essentially, a larger K allows for better local deep statistics in the deep context feature, which enables better kernel predictions. Note that this is not the case for standard PM using the same nearest neighbor strategy for bandwidth selection. Photon mapping either introduces noticeable non-smooth artifacts with a small bandwidth (Figure 2.8(i)) or outputs over-smooth results without details with a large bandwidth (Figure 2.8(d)). APPM tends to resolve this issue by wisely reducing the bandwidth according to the photon statistics. In contrast, our method achieves significantly better results than APPM when there are only sparsely distributed photons. Our method can leverage a relatively large bandwidth without introducing any obvious over-smoothing issues. This is thanks to our learning-based context-aware kernel prediction approach. In particular, our approach allows for every single photon to leverage across-photon information in the learned deep context feature to tell if it is an outlier or an important contributing element to the shading point's reflected radiance; a corresponding kernel weight is assigned to each photon based on the decision made by data priors in the network. Therefore, our method can effectively utilize the sparse photons in a large area to generate photorealistic images that are of high smoothness and have many details.

2.8.6 Timing and Overhead

We use Optix to trace photons and do path tracing for all the results. All experiments are run on one NVIDIA 1080 Ti GPU. Path tracing runs at about 50 spp per second in all six scenes with an image resolution of 512×512. It takes about 0.1, 1.0, and 10 seconds to emit photons. We show the corresponding photon gathering time and network inference time for 512×512 surface shading points in Table 2.2. In particular, we build Kd-Trees to do the neighboring search at each shading point, and all methods take a similar time to gather neighboring photons. Note that the running time of our network is linear with the number of input photons K; it is also determined by the number of shading points that are required to be computed, and the listed timing corresponds to 512×512 shading points. The total running time for our method is the summation of the photon tracing, gathering, and network inference time; the total running time for the other methods is the summation of tracing and gathering. Note that, across all the experiments (Table 2.1, Figure 2.8, Figure 2.9), our results of DPM-500 with photons traced in 1 second are comparable to the best results of comparison methods with photons traced in 10 seconds; however, to achieve the comparable results, our DPM-500 takes about 5.2s $\sim$ 9s total time, whereas the comparison methods require 22s $\sim$ 60.0s total time to compute the same number of shading points. Our method takes a significantly shorter time to achieve comparable quality.

2.8.7 Effect of Variable Attributes

While our network is mainly trained with relatively sparse photons (small M), our network with K=500 overall generalizes well across different numbers of total photons (M) and, in most cases, achieves better performance when M increases. However, for K=50, there is too little information for the network to leverage and higher performance is often not ensured with a larger M. Nonetheless, our network with K=50 still works well and performs better than the comparison methods when there are tens of thousands of photons. We also observe that a larger network

Table 2.3. Temporal stability. We show the mean DSSIM between pairs of adjacent frames over a sequence of 30 rendered frames. Results have been averaged over all the different scenes and amount of photons.

	Ours-50	Ours-Large-50	PM-50	Ours-500	Ours-Large-500	PM-500
Mean DSSIM	0.0346	0.0342	0.0337	0.0281	0.0277	0.0260

(Ours-L) with a larger capacity leads to clearly better results than our regular network. Of course, a larger network requires a higher computational cost or longer inference time, as shown in Table 2.2. Nevertheless, the larger network with $K = 50$ can already often achieve reasonably good results, which takes a shorter running time than $K = 500$. We leave the exploration of more variants of the network capacity and K as future work.

2.8.8 Temporal Consistency

Since our method deals with shading points in 3D space and is independent of view directions, we have observed good across-frame consistency when changing the view in a scene with a fixed set of photons. We follow [172] and use the mean DSSIM between consecutive frames to evaluate the temporal consistency when moving the camera. Results in Table 2.3 show comparable temporal stability between our results and standard PM outputs. We leave the extensions of our network to recurrent architectures and general temporal consistency with other dynamic components in a scene as future work.

2.8.9 Progressive Density Estimation

Our current framework requires a fixed number of input photons for each trained network. Progressive photon mapping accepts different numbers of photons per iteration with reduced bandwidth. Nonetheless, we have demonstrated that our network architecture supports accurate photon density estimation with various fixed photon numbers. In other words, a progressive method can potentially be achieved by training a sequence of networks with different numbers of inputs. A universal network for any given number of input photons may require introducing recurrent networks in the framework, which is an interesting direction of future work.

2.9 Summary of Contribution

In this chapter, we have presented the first deep learning-based method for density estimation in particle-based rendering. We introduce a deep neural network that learns a kernel function to aggregate photons at each shading point and renders accurate caustics with significantly fewer photons than previous approaches, with minimal overhead. Learning-based MC denoising has significantly improved path tracing results, and our work extends these benefits to the popular photon mapping method.

Our method could be improved in the future with more advanced machine learning approaches, perhaps based on generative adversarial networks (GANs), just as has been done with path tracing [193]. More broadly, we believe this work points towards denoisers specialized to many other approaches for realistic image synthesis such as Metropolis Light Transport and Vertex Connection and Merging.

Chapter Review. In this chapter, we have examined technical challenges in creating caustics effect on images. We then introduced a new radiance reconstruction framework grounded on particle-based rendering to reduce the computational cost.

Acknowledgements. This chapter, in full, is a reprint of the material as it appears in Eurographics Symposium on Rendering 2020 and Computer Graphics Forum 2020 [212]. Shilin Zhu, Zexiang Xu, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi, Wiley-Blackwell, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Computing Universal Illuminations on Images

3.1 Generalized Global Illumination

The particular caustics effect in Chapter 2 is one type of *global illumination* where lights are redirected to concentrated areas. Generally speaking, depending on the total number of scattering events, we can categorize the lighting into direct and indirect components, as shown in Figure 3.1. They both contribute significantly to the realistic appearance and must be synthesized accurately to enable universal effects on images.

Typically, the direct illumination is simpler to compute thanks to shorter paths. On the other hand, the multi-bounce nature of the indirect part makes the light transport exceedingly difficult to discover all inter-reflected paths. According to the theory of Monte Carlo, applying a superior sampling strategy (e.g., importance sampling as demonstrated in Figure 3.1) increases the chance of constructing valid transmission paths. The design of sampling distributions for the global lighting effect is the primary focus of this chapter containing two published articles [214, 215].

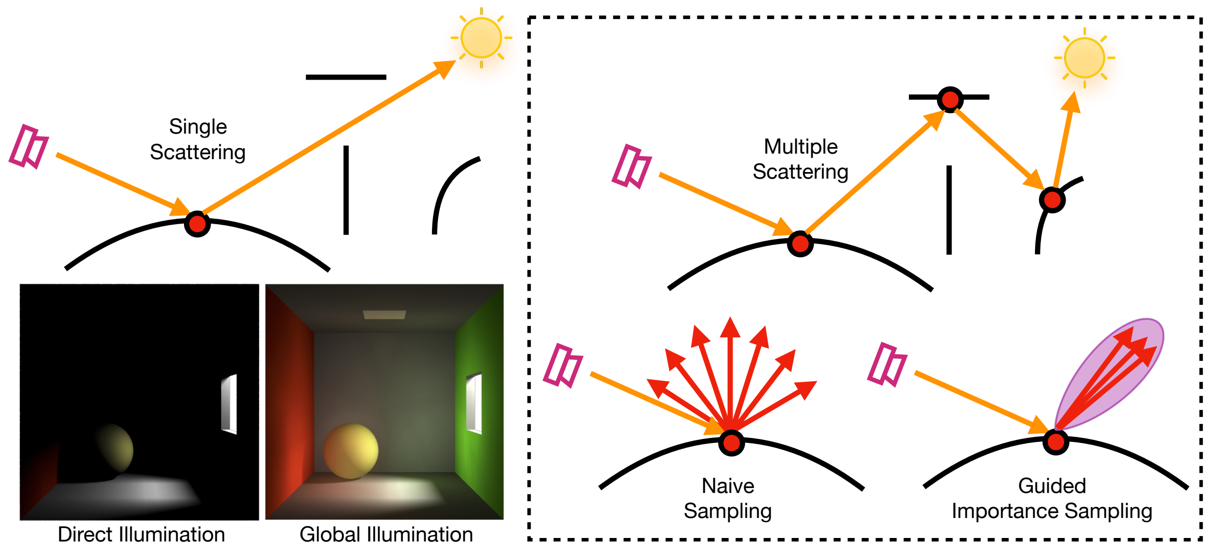


Figure 3.1. Illustrations of global illumination and importance sampling. The generalized image synthesis task involves simulating effects from any type of scattering event. In most cases, the indirect component is more challenging to compute due to the inefficient sampling tactics. As a result, it is crucial to build guiding distributions for driving rays toward light sources.

3.2 Related Work

We have conducted two research related to path sampling. Our initial attempt [214] (Section 3.3) is regarded as one of the prior arts from the perspective of the most recent advancement [215] (Section 3.4).

Path Guiding. Monte-Carlo path tracing [81] has been the fundamental solution for solving the light transport in a complex scene. However, during path tracing, the incident light distribution is unknown at each 3D point. Thus, most path tracing variants sample the space only based on the geometry and reflectance properties. Instead, path guiding algorithms [176] estimate sampling distributions based on the local incoming light field during path tracing, so that they can use the information to perform better importance sampling and accelerate the rendering process.

Several path guiding algorithms (Table 3.1) have been proposed to efficiently estimate the local light field information in order to sample the space better. Vorba et al. [175] fitted a Gaussian-Mixture model (GMM) to represent the incoming radiance at each spatial cache point

Table 3.1. Comparison of different path guiding algorithms. Our proposed framework can achieve both fast and robust rendering by leveraging neural networks and hybrid samples with a small memory consumption thanks to the hierarchical representation of sampling distributions.

	Hybrid	Hierarchical	Neural
[175]	✗(Photon)	✗(GMM)	✗
[111]	✗(Path)	✓(Quadtree)	✗
[110]	✗(Path)	✓(Quadtree)	✗
[6]	✗(Path, 1st bounce)	✗(Image)	✓
[132]	✗(Path)	✓(Quadtree)	✗
[214]	✗(Photon)	✗(Image)	✓
Ours	✓(Path + Photon)	✓(Quadtree)	✓

during ray tracing. With very few parameters, a GMM can efficiently model the light distribution, and is then applied to other rendering algorithms [65, 138]. However, GMMs fail to accurately represent high-frequency light distributions, which are common in scenes with complex lighting. Müller et al. [111, 112] proposed to use hierarchical quadtree structures to record the incoming light field in the space, which is more efficient and practical than a GMM [175] or simple regular grid [76]. This hierarchical representation was also extended to primary space [58], product sampling [27], and variance-aware importance sampling [132]. However, until now, such a hierarchical representation can only be reconstructed via traditional online learning without any neural network components, and requires a relatively large number of samples. Our neural approach can directly reconstruct an accurate hierarchical quadtree representation from sparse input samples using an offline-trained novel deep neural network.

Recently, deep learning techniques have been used to facilitate the learning of local light distributions and importance sampling of light paths (e.g., in primary sample space [207]). Müller et al. [112] used an online-learned neural network to perform the importance sampling. The network can estimate the distribution accurately, but can be potentially expensive in practice due to the repeated network inference and online optimization. Bako et al. [6] trained an offline-learned network to guide the first bounce, where regular images are used to represent the incoming light distribution. While images are convenient for neural networks, they consume more memory when detailed light distributions are needed. Huo et al. [71] used a reinforcement learning technique to guide the samples, but their method is also limited to the first bounce. Zhu et al. [214] used photons as the primary source to estimate the local light distributions, and use them to guide all bounces. Again, standard images are used to represent the distributions, which are less memory efficient and limited to low resolutions compared to the quadtree. In this work, we learn the light distribution on hierarchical structures, which are both detailed and memory-efficient. Our approach takes advantage of using both path and photon samples, leading to better generality on different scenes. We believe these are important steps to make neural path guiding practical.

Hierarchical Learning. Hierarchical structures can represent sparse data in an efficient way [111, 110]. However, learning on hierarchical structures has been a particular challenge. Recently, plenty of studies have emerged to focus on the learning and understanding of hierarchical structures, especially in the 3D geometry processing community. Wang et al. [181, 182] proposed O-CNN to analyze 3D shapes represented by octrees; Graham [53] developed sparse convolution for 3D understanding, which is similar to sparse matrix representation. On the other hand, there are also works on generating hierarchical structures [157, 19, 135]. These algorithms were then extended to perform 3D shape completion [183], 3D segmentation [54, 55], and sketch understanding [91]. Besides convolutional operators, multi-layer perceptrons [96, 97] and graph networks [105] are also used for hierarchical learning. In this work, we extend these hierarchical 3D learning techniques to the problem of 2D sampling distribution reconstruction. We introduce a novel lightweight network that can effectively regress an accurate quadtree distribution for high-quality path-guiding.

Hybrid Samples. Both paths and photons are efficient tools to explore the scene and compute the radiance in the 3D space. While path tracing [81] algorithms are particularly good at exploring complex geometry setups, photon mapping algorithms [145, 77, 61, 89, 212] can be very effective when indirect lighting dominates the scene. Aiming at a rendering algorithm that can work on both cases, researchers proposed bidirectional approaches [93, 170, 46, 90], which combine the benefits of both path tracing and photon mapping. Similarly, in our case, we use both path samples and photons as the sources to learn the local light distributions in the scene. Compared to the previous path guiding works that use only path samples [111, 6, 132] or photons [175, 214], our algorithm can render more efficiently and is more robust across a wide range of difficult scenes with complex light transports. In fact, Vorba et al. [175] also use both path and photon samples. However, they train with two separate cache records, where path tracing is guided by local photons and path samples do not *directly* affect camera path guiding. Our neural system instead takes the *hybrid* of two types of samples as direct inputs; they directly contribute to the same forward sampling distribution.

As shown in Table 3.1, our path guiding algorithm uniquely utilizes the hybrid samples. Additionally, previous works either perform learning on image-based sampling distributions, or use hierarchical structures to represent the distributions (because of the difficulty of applying neural networks to irregular quadtree structures), but not both. In contrast, our path guiding algorithm successfully applies an offline-learned neural network on hierarchical structures.

3.3 The First Attempt: Photon-Driven Path Guiding

Although Monte Carlo path tracing is a simple and effective algorithm to synthesize photo-realistic images, it is often very slow to converge to noise-free results when involving complex global illumination. One of the most successful variance-reduction techniques is path guiding, which can learn better distributions for importance sampling to reduce pixel noise. However, previous methods require a large number of path samples to achieve reliable path guiding. We present a novel neural path guiding approach that can reconstruct high-quality sampling distributions for path guiding from a sparse set of samples, using an offline trained neural network. We leverage photons traced from light sources as the primary input for sampling density reconstruction, which is effective for challenging scenes with strong global illumination. To fully make use of our deep neural network, we partition the scene space into an

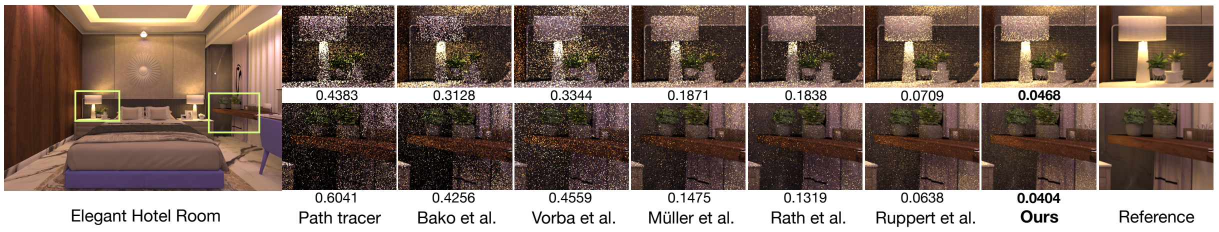


Figure 3.2. We present a novel photon-driven neural path guiding approach that can effectively reduce the variance in path tracing. This complex scene is lit by several decorative lights which are very difficult to discover in path tracing. We compare the equal-time ($\sim$20 minutes) rendering results with standard path tracing and state-of-the-art path-guiding methods (including [111], [6], [132], and [138]), showing the crops (illuminated mostly by the *lamp lights*) of the rendered results with corresponding relative MSEs (rMSEs). [6] use an offline trained neural network for path guiding; however, it only supports guiding the first bounce, which is not very effective since this scene is dominated by indirect lighting. While traditional methods allow for multi-bounce path guiding, they are mostly online learning methods and they need longer time to learn the complex sampling functions for this challenging scene. Our method utilizes a trained deep neural network and enables effective path guiding at any path bounces.

adaptive hierarchical grid, in which we apply our network to reconstruct high-quality sampling distributions for any local region in the scene. This allows for effective path guiding for arbitrary path bounce at any location in path tracing. We demonstrate that our photon-driven neural path guiding approach can generalize to diverse testing scenes, often achieving better rendering results than previous path guiding approaches and opening up interesting future directions.

3.3.1 Problem Definition and Contribution

Monte Carlo path tracing has been widely used in photo-realistic image synthesis. However, while simple and flexible, path tracing can take a significant amount of time to generate noise-free images for complex scenes (e.g., Figure 3.2). One critical challenge for Monte Carlo based methods is to effectively construct light transport paths connecting the light and the camera.

Many path-guiding methods [111, 76] have been presented to construct advanced distributions (usually approximating incident light fields or some variants of those) for importance sampling at local shading points, guiding the local path sampling for high-energy path construction. The recent successful methods are based on unidirectional guiding [111, 132, 138]; they rely on early path samples to discover high-energy sampling directions. Although the strategy is generally effective, this unidirectional path discovery process can still be slow for a challenging scene dominated by indirect illumination. While using light paths is known to be efficient in exploring the path space, previous photon-driven or bidirectional path-guiding methods [76, 175] are not yet efficient, requiring sampling a large number of light paths.

We present a novel path guiding approach that can achieve effective path sampling using *a sparse set* of light paths as input, thus successfully advancing the overall rendering speed. Inspired by the original path guiding work [76], we leverage photons to compute local sampling distributions for importance sampling in path tracing, where a sampling distribution at any 3D local region can be obtained by binning local photons according to their directions (i.e., a 2D histogram map). However, such distributions are only reliable with locally dense photons, are usually low-quality, and appear noisy with sparse photons (Figs. 3.3 and 3.4).

We propose to use a compact neural network to reconstruct high-quality sampling distributions for path guiding from low-quality noisy histograms that are acquired by binning sparse photons. In essence, we break down the complex path guiding problem, mainly focusing on reconstructing local sampling distributions represented as 2D maps (i.e., images), and thus pose this problem as one of the image-to-image translations that can now be addressed by deep learning techniques. Our neural reconstruction network is effectively trained offline in a scene-independent way. It can recover the shapes of complex sampling distributions on new scenes, enabling guided path tracing with complex global illumination effects.

Our framework is designed to reconstruct high-quality sampling maps at local spatial regions. To make these sampling maps well distributed and locally representative in the scene space, we adaptively partition the entire scene space into a hierarchical grid, according to the complexity of local incident light variations. The neurally reconstructed sampling maps are cached in leaf voxels of the grid, enabling path guiding at different locations in a scene. Therefore, we can support guiding path tracing at multiple bounces. Although our approach also has specific limitations (e.g., reconstructing only low-resolution sampling maps because of the memory limit, experiencing uneven photon visibility), we demonstrate that our novel learning-based path guiding often achieves better rendering quality on various challenging scenes than previous state-of-the-art path-guiding methods when GPU resources are available (Figure 3.2). The proposed reconstruction framework serves as a starting point for many extension possibilities. In summary, our main contributions are:

- We present a learning-based approach that leverages photons to reconstruct high-quality sampling distributions locally;
- By combining with an adaptive spatial caching structure, we support building and reusing better sampling distributions at arbitrary bounces to reduce variance of path tracing results.

3.3.2 Introduction to Importance Sampling

Physically-based rendering can be expressed by the Rendering Equation [81] that describes the radiance leaving an intersection point $\boldsymbol{x}$ in direction ω_o:

$$L(\boldsymbol{x}, \omega_o) = L_e(\boldsymbol{x}, \omega_o) + \int_{\Omega} L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i d\omega_i, \tag{3.1}$$

where $L_e(\boldsymbol{x}, \omega_o)$ denotes the emitted radiance, $L_i(\boldsymbol{x}, \omega_i)$ is the incident radiance from direction ω_i, f_r is the bidirectional scattering distribution function (BSDF), and Ω corresponds to the full sphere. The key component in the equation is the integral that computes the reflected radiance $L_r(\boldsymbol{x}, \omega_o) = \int_{\Omega} L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i d\omega_i$ over all directions in the sphere.

The integral can be numerically evaluated using Monte Carlo estimation [169]:

$$L_r(\boldsymbol{x}, \omega_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)} \tag{3.2}$$

where N Monte Carlo path samples in various directions ω_i are drawn from the probability density function (PDF) $p(\omega_i)$. Considering global illumination with multiple bounces, $L_i(\boldsymbol{x}, \omega_i)$ is computed by recursively evaluating integrals using Equation 3.1. In path tracing, rays are sampled from each intersection point to compute the radiance that contributes to the pixel color at multiple bounces.

The variance of the Monte Carlo estimate $L_r(\boldsymbol{x}, \omega_o)$ can be reduced by sampling ω_i from a density function $p(\omega_i)$ that resembles the numerator $L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i$. Ideally, if $p(\omega_i)$ and the numerator only differ by a constant scale, the variance is reduced to zero. However, this numerator is unknown and is as difficult as the integral to compute, due to complex visibility and indirect lighting in L_i; therefore, standard path tracing often proceeds with BSDF importance sampling for indirect lighting plus a direct light sampling technique (e.g., next-event estimation).

Path guiding aims to reconstruct a density function that matches the shape of the numerator as closely as possible. In particular, since the standard BSDF importance sampling satisfies [169]:

$$p_{\text{BSDF}}(\omega_i) \propto f_r(\boldsymbol{x}, \omega_i, \omega_o) \tag{3.3}$$

recent path-guiding methods often set the target probability density to be proportional to the incident light [175, 111, 138] (the following cosine term is sometimes included in BSDF sampling in Equation 3.3 instead of guiding):

$$p_{\text{guide}}(\omega_i) \propto L_i(\boldsymbol{x}, \omega_i) \cos\theta_i. \tag{3.4}$$

The final sampling strategy is achieved by combining the guiding and BSDF sampling using either the product sampling (i.e., $p_{\text{guide}}(\omega_i) \cdot p_{\text{BSDF}}(\omega_i)$ [65]) or the one-sample Multiple Importance Sampling (MIS): [171]

$$p(\omega_i) = \alpha p_{\text{BSDF}}(\omega_i) + (1-\alpha) p_{\text{guide}}(\omega_i), \tag{3.5}$$

where α is the mixture coefficient that determines the probability of choosing BSDF sampling or guided sampling.

Many recent works rely on early path samples in path tracing to approximate the incident

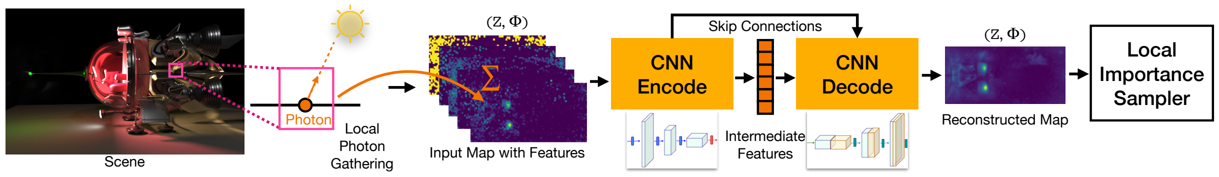


Figure 3.3. Illustration of the entire system workflow. We partition the scene into multiple local regions using a spatial structure, where each voxel gathers neighboring photons locally. The gathered photon statistics are accumulated in the directional space represented by a noisy histogram map with additional features, and photons are deleted after such splatting is completed during particle tracing. Next, we use a pre-trained compact CNN to encode the input map to neural features, followed by a decoder to reconstruct a target sampling map. The output map is reused by a local importance sampler to decide the next sampled direction at any bounce in path tracing.

light field (Equation 3.4), which is insufficient for challenging scenes with strong indirect lighting (Figure 3.2). We instead leverage photons traced from the lights to compute the sampling density functions, which effectively explores the challenging light transport. Our novel approach advances the traditional path guiding with powerful deep learning techniques and an adaptive spatial structure, enabling effective path guiding from sparse photons.

3.3.3 Workflow Overview

Our path guiding approach uses a compact pre-trained neural network to regress high-quality sampling maps that can be used to guide path sampling. Meanwhile, we utilize an adaptive hierarchical grid for spatially storing the reconstructed distributions, enabling effective path guiding at multiple bounces. The whole system is illustrated in Figure 3.3.

In the following sections, we first introduce our sampling map parameterization, target sampling density, and how to use photons to compute the histograms in Section 3.3.4. We then introduce our deep neural network that can regress better sampling maps given noisy low-quality histograms in Section 3.3.5. We present our full path guiding framework in Section 3.3.6, which describes our iterative sample gathering and rendering process, adaptive spatial structure, and how paths, photons, and the neural network are incorporated into the system.

3.3.4 Computing Sampling Maps

Previous methods [76, 175] usually compute hemispherical distributions at sampled surface points to approximate incident light fields. However, such hemispherical functions approximate light fields at locally flat 2D surface regions, and are hard to interpolate on surfaces with complex normal variations. Inspired by the recent unidirectional path-guiding methods [111, 132, 6], we utilize a full spherical sampling distribution that models the incident light distribution in a local 3D region. In particular, we build a hierarchical grid (Section 3.3.6) in the scene space, and compute a spherical sampling distribution stored in a discrete data structure for each local 3D voxel of the adaptive grid. In this section, we discuss the representation of our sampling function and its computation from photons during particle tracing from light sources.

Spherical Function Representation. We use a regular directional grid that represents the sampling density function as a 2D sampling map (similar to [6]). We leverage the cylindrical mapping to parameterize the spherical domain for better area preservation (similar to [111, 132]). In particular, a vector $r = (x, y, z)$ on a unit sphere is mapped to a 2D location $(u, v) = (z, \phi)$ on the sampling map, where $\phi = \arctan(y/x)$. This sampling map is similar to a standard environment map or radiance map in traditional lighting representation, but ours is monochromatic and uses cylindrical parameterization.

Target Sampling Density. As discussed in Section 3.3.2 (Equation 3.4), the goal of path guiding is to compute the sampling density at any position, making it proportional to the incident light $L_i(\boldsymbol{x}, \omega_i)$ or $L_i(\boldsymbol{x}, \omega_i) \cos\theta_i$. For our discrete case where we consider a 3D voxel region and a certain footprint (representing a solid angle bin) of a sampling map, it is in fact the expected incident light that is of our interest. In particular, given a voxel j and a solid angle footprint $\Delta\Omega_k$ of pixel k in the sampling map, the expected $L_i(\boldsymbol{x}, \omega_i) \cos\theta_i$ coming from the solid angle over the local surface area ΔA_j (that is of the scene geometry covered by the voxel) is expressed by:

$$\mathbb{E}[L_i(\boldsymbol{x},\omega_i)\cos\theta_i] = \frac{\int_{\Delta A_j}\int_{\Delta\Omega_k} L_i(\boldsymbol{x},\omega_i)\cos\theta_i d\omega_i d\boldsymbol{x}}{\Delta\Omega_k \Delta A_j} \tag{3.6}$$

$$= \frac{\Phi_{j,k}}{\Delta\Omega_k \Delta A_j}, \tag{3.7}$$

where $\Phi_{j,k}$ represents the total incident power in the spatial and directional range. Therefore, it is the total power (radiant flux)

$$\Phi_{j,k} = \int_{\Delta A_j}\int_{\Delta\Omega_k} L_i(\boldsymbol{x},\omega_i)\cos\theta_i d\omega_i d\boldsymbol{x}, \tag{3.8}$$

that governs our sampling map distribution. Essentially, $\Phi_{j,k}$ models the integrated incident radiance. Note that the irradiance ($E(\boldsymbol{x},\Delta\Omega_k) = \int_{\Delta\Omega_k} L_i(\boldsymbol{x},\omega_i)\cos\theta_i d\omega_i$) is a standard radiometry term and widely used in previous works [76, 132]; when divided by the surface area, $\Phi_{j,k}$ also describes the expected irradiance ($\Phi_{j,k}/\Delta A_j$) in the voxel. Therefore, we seek to obtain sampling densities that are proportional to the expected incident light:

$$p_{\text{guide}}(\omega_i) \propto \Phi_{j,k}/\Delta\Omega_k, \tag{3.9}$$

where we ignore the ΔA_j in Equation 3.7 since it is constant for all solid angles in the same voxel. This sampling density corresponds to a sampling map, each pixel value of which is proportional to $\Phi_{j,k}$. We thus reconstruct a sampling map by normalizing a power map that records the power $\Phi_{j,k}$ in each pixel.

Computing Incident Illumination. In this work, we leverage particle tracing to effectively evaluate the $\Phi_{j,k}$ (Equation 3.8). We trace light paths from the light sources to distribute photons in the scene, where each photon carries a portion of flux; $\Phi_{j,k}$ can then be estimated by simply binning the photons similar to [76]:

$$\Phi_{j,k} = \sum_{\omega_p \in \Delta\Omega_k, \boldsymbol{x}_p \in \Delta A_j} \Delta\Phi_p, \tag{3.10}$$

where p denotes a photon arriving at the surface point $\boldsymbol{x}_p$ from direction ω_p, and $\Delta\Phi_p$ is the power carried by it. Equation 3.10 essentially accumulates all the photon power inside a voxel and directional bin.

Note that Müller et al. [111] leverages path tracing to accumulate the radiance samples inside a local voxel; this can be seen as an integral of the radiance over an area and a solid angle, similar to our power expression Equation 3.8. Our particle-based approach provides an unbiased estimate for the power integral $\Phi_{j,k}$ when the photon count goes to infinity.

Since the evaluation is governed by accumulating splatted values to a histogram map, we can progressively trace as many photons as needed without storing the entire photon point cloud (required by traditional photon mapping, leading to memory bottleneck from more traced particles). Once a photon is accumulated to a directional bin of the map inside a voxel, it is then deleted, except for the initialization phase (Section 3.3.6). Note that an accurate power map requires tracing a large number of photons, but in practice, we can only allow for tracing a small number of photons at rendering time, which by themselves cannot directly lead to high-quality sampling.

3.3.5 Neural Reconstruction of Sampling Distributions

If directly computing sampling distributions by binning photons, neither dense photons (slow) nor sparse photons (low-quality) are suitable for efficient path guiding. To overcome this, our central idea is to obtain accurate sampling maps offline as ground truth using dense photons, and leverage supervised learning to regress such maps from low-quality histograms that can be computed efficiently from sparse photons. Specifically, we propose to train a deep Convolutional Neural Network (CNN) that learns to reconstruct a high-quality sampling distribution from sparse photons.

Our sampling maps are reconstructed and updated repeatedly through multiple iterations in our path guiding framework (Section 3.3.6). We consider a normalized noisy sampling map S_t as input, acquired by accumulating a sparse set of photons from iteration 1 to t using Equation 3.10, where t denotes the iteration number. We also supply the noisy sampling map S_{t-1} from the previous iteration to ease the learning of where to in-paint. In addition, we record the number of photons per solid angle bin in S_t and S_{t-1}, resulting in maps P_t and P_{t-1}, and include the normalized buffers in the input. Inspired by the image inpainting techniques [101, 201, 199], we also concatenate a binary mask B_t indicating whether a solid angle bin contains photon data or not, and use light-weight masked convolutions to process the input maps. As a result, our full input is an image map with 5 separately-normalized (by the GPU) channels and our neural network $\mathbb{F}$ can be expressed by:

$$S_d = \mathbb{F}(S_t, S_{t-1}, P_t, P_{t-1}, B_t). \tag{3.11}$$

The output is a one-channel sampling map S_d (which is then converted to CDF for importance sampling), supervised by the ground-truth map $\tilde{S_d}$ computed from dense enough photons.

Neural Architecture and Loss Computation. Our network is essentially designed to solve an image-to-image reconstruction task. Many existing 2D neural networks for image-to-image denoising, translation, and inpainting ([15, 5, 172, 101]) can thus be potentially applied to address the problem. However, our neural network is applied on a large number of voxels, while our end goal is to speed up the overall rendering process. Therefore, we balance the inference speed and reconstruction quality in the network design.

We propose to use a compact U-Net [136] style architecture with residual links and skip connections to achieve the sampling map reconstruction as illustrated in Figure 3.3. It contains multiple downsampling and upsampling convolutional layers to extract high-level neural features from the input map S_t and output a better version S_d. The input noisy sampling maps are computed from sparse photons, which contain many empty bins. Therefore, we use the light-weight masked convolutions inspired by the recent image inpainting works [101, 199], which ensures that valid (non-empty) and invalid (empty) bins are treated differently and only valid bins can contribute to convolutions. Note that the designed architecture is relatively compact compared to the previous deep networks ([15, 5, 172]) used in denoising. Although the extra computational overhead introduced by the neural network is inevitable, the compactness allows sampling map reconstruction to finish in a reasonable time on powerful GPUs. Due to limited system memory, we reconstruct low-resolution maps (64×128 or 32×64), which are already adequate for path guiding in typical scenes. We believe our architecture can be further improved by advanced compression techniques [18, 25] and novel neural components, and we leave this as future work.

We utilize the standard L_1 loss to supervise the output sampling map:

$$\mathbb{L}_S = |\hat{S}_d - S_d| \tag{3.12}$$

where $\hat{S}_d$ is the ground-truth sampling map computed by accumulating dense photons. Inspired by the deep supervision [192, 94], we also provide the ground-truth signal on each decoding level to ease the loss back-propagation. To prevent over-blurring, we leverage an asymmetric function inspired by Vogels et al. [172]; this leads to our full loss

$$\mathbb{L}_{\text{rec}} = \mathbb{L}_S \cdot (1 + (\lambda - 1) \cdot \mathbb{H}) \tag{3.13}$$

where $\mathbb{H} = 0$ if the output and the input values are both larger or smaller than the ground-truth value and $\mathbb{H} = 1$ if they are not on the same side. Specifically, when there are two equally-good output values, the function prefers the one that is closer to the input. This allows the reconstruction to retain some noise but also prevent details from being blurred out.

In Figure 3.4, we present some examples of the reconstructed sampling maps along

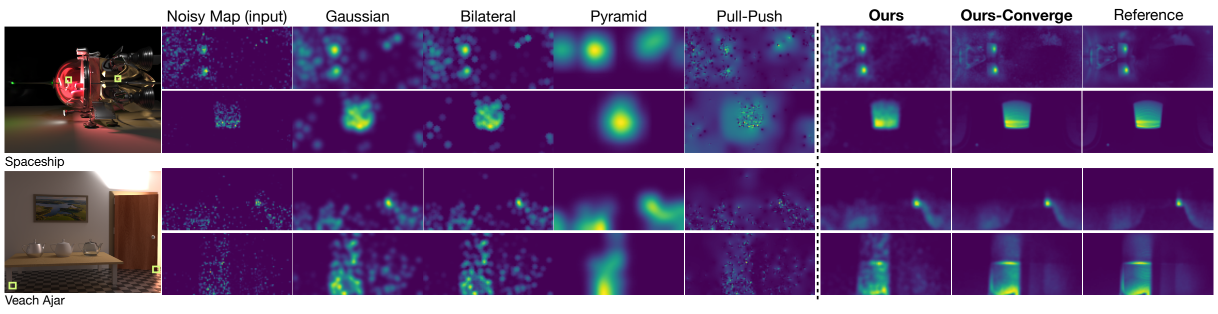


Figure 3.4. Example reconstructed sampling maps (gamma transformed for better visualization purpose). With more iterations of path and photon tracing, the reconstructed sampling map can converge to the reference (after 8 iterations in this example). We also compare to other traditional image interpolation techniques (3 hierarchical levels for pyramid and pull-push filters). Although they can also improve the quality of the noisy histogram maps through blurring in general, our neural-based reconstruction is designed specifically for this task, thus producing maps closer to the targets.

with the comparison with other traditional image inpainting techniques. We can clearly see the advantage of a group of learned filters represented by a data-driven neural network trained specifically under the context of path guiding, over the traditional hard-coded filters designed for general image processing (not specifically for path guiding) based on human knowledge.

Applicability. The proposed neural network focuses on reconstructing high-quality distributions for local path sampling. This is a central sub-problem in many path guiding frameworks. Note that the problem of sampling map regression is independent of other modules in path guiding. We thus train our neural network independently without relying on any specific guiding frameworks. Therefore, our learning-based sampling map reconstruction module can potentially be extended to other existing path guiding frameworks by applying a proper variant of our neural architecture (e.g., mixture models [175, 138] and hierarchical maps [111, 132]), and improves the traditional sampling distribution reconstruction modules.

3.3.6 Adaptive Path Guiding Framework

In this section, we introduce our path guiding framework that leverages the presented deep network to reconstruct high-quality sampling maps in an adaptive and hierarchical spatial structure. The whole framework is illustrated in Algorithm 1.

ALGORITHM 1: Our neural path guiding framework. Through multiple iterations of path and light tracing, we construct a hierarchical grid (in green), reconstruct and update the sampling map in each valid voxel (in blue), and guide the path tracing using the reconstructed distributions (in red). We also apply a final guided path tracing pass (in purple).

```
Initialize 1 SPP path samples and 1 SPP photons ;
Initialize a spatial grid ;
for each iteration $t < T$ do
    Initiate $2^t$ SPP path samples;
    for each path do
        for each bounce $b$ do
            Locate voxel $j$ ($\boldsymbol{x}_b \in \Delta A_j$) ;
            if not isValid($j$) (no sampling map) then
                Sample($p_{\mathrm{BSDF}}$) $\rightarrow \omega_b$ ;
            else
                Sample($p_{\mathrm{MIS}}$) $\rightarrow \omega_b$ (Equation 3.15);
            end
            markValid($j$) ;
        end
        Compute path throughput and $L(\boldsymbol{x}_b, \omega_b)$ ;
        for each bounce at $\boldsymbol{x}_b \in \Delta A_j$ do
            if isValid($j$) then
                $L_b = L(\boldsymbol{x}_b, \omega_b) \cos\theta_b f_r(\boldsymbol{x}, \omega_b, \omega_o)$ ;
                if $\omega_b \leftarrow p_{\mathrm{guide}}$ then $L_{j,\mathrm{Guide}} \mathrel{+}= L_b$ else $L_{j,\mathrm{BSDF}} \mathrel{+}= L_b$ ;
                if $\omega_b \leftarrow p_{\mathrm{guide}}$ then $Q_{j,\mathrm{Guide}} \mathrel{+}= 1$ else $Q_{j,\mathrm{BSDF}} \mathrel{+}= 1$ ;
                if $Q_{j,\mathrm{Guide}} \geq Q_{\mathrm{Thr}}$ & $Q_{j,\mathrm{BSDF}} \geq Q_{\mathrm{Thr}}$ then Update $\alpha_j$ (Equation 3.14);
            end
        end
        Update the output image ;
    end
    Trace $2^t N_p$ light paths for photons;
    for each photon $p$ do
        Locate voxel $j$, solid angle $k$ ($\boldsymbol{x}_p \in \Delta A_j$, $\omega_p \in \Delta\Omega_k$) ;
        if isValid($j$) then
            Update power map: $\Phi_{j,k} \mathrel{+}= \Delta\Phi_p$ (for Equation 3.10);
            $M_j \mathrel{+}= 1$ ;
            if $M_j > M_{thr}$ then
                Subdivide voxel $j$ into two sub-voxels (Section 3.3.6);
            end
        end
    end
    for each valid voxel $j$ do
        Reconstruct sampling maps ($p_{\mathrm{Guide}}$) with neural net $\mathbb{F}$ ;
    end
end
Trace $N_f$ paths for final output (Section 3.3.6);
```

We first fire some initial path and light rays, initialize a grid, and then utilize an iterative process to adaptively build a hierarchical grid with per-voxel sampling maps for path guiding and rendering.

In each iteration, we trace camera paths; these paths can be guided when tracing, and they are used to detect valid (i.e., containing path vertices) voxels and compute the mixture weight of one-sample MIS. We also trace photons per iteration; in each valid voxel, we accumulate photon power that is required by our neural module and collect photon statistics for subdividing the hierarchical grid. We then reconstruct the sampling map in each valid voxel using our pre-trained deep neural network at the end of each iteration; these sampling maps are used to guide the path tracing in the next iteration. After the iterative process, we apply a final-pass guided path tracing and compute the final beauty image.

We adaptively partition the scene space to a hierarchical grid. Meanwhile, the photons are collected for computing the noisy sampling maps in each valid voxel; the path samples are used for rendering and computing the weight α for one-sample MIS. After T iterations, we run a final path tracing pass with N_f spp. The final rendering result is computed by combining all iterations (except for the initialization phase) and the final pass, weighted by the inverse of their estimated variances [110]. Note that we double the number of path and photon rays ($2^t N_c$ and $2^t N_p$ spp for iteration t, where N_c and N_p are initial values) after each iteration [111], so that both the quality of the input noisy histograms and per-pass rendering can be progressively improved.

Adaptive Hierarchical Grid Caching. Recent work often utilize a binary KD-Tree [111, 132] to adaptively partition the space, starting from a single root node that covers the entire scene. This coarse-to-fine spatial structure is effective and also necessary for these online learning approaches, since they need to acquire many samples in a large spatial region at an early stage. In contrast, our deep learning-based approach can reconstruct a high-quality sampling map from a sparse set of photons; consequently, starting from a single root node is unnecessary and inefficient for our approach. Therefore, we propose to use a hierarchical grid for spatial partitioning, which combines uniform and adaptive spatial partitioning (Figure 3.5).

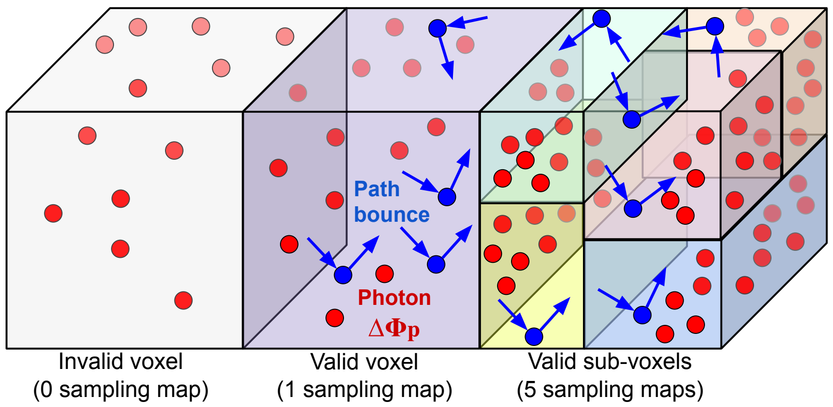


Figure 3.5. The proposed hierarchical grid spatial caching structure. Path samples detect valid voxels to store sampling maps. A voxel is subdivided into a binary tree based on the local photon statistics. We split through the median to prevent skew or distorted distributions reconstructed in the initialization phase, and switch to the middle plane splitting in the iterative update phase when the photon point cloud is no longer stored. We alternate the splitting dimension with respect to the tree depth.

Detecting Valid Voxels. While we can compute a sampling map for every voxel for path guiding, this is usually costly and unnecessary, since many voxels may not be reached by any camera path from the viewpoint. Therefore, we leverage camera paths to detect which voxels are involved in rendering this viewpoint. Specifically, when tracing $2^t N_c$ spp path samples in each iteration, we mark a voxel as valid if there is at least one bounce point of the paths located in the voxel (Figure 3.5). Once a voxel is marked as valid, we then start accumulating photons in the voxel for sampling map reconstruction and further subdivision of the voxel. This avoids the waste of caching redundant distributions and local KD-trees.

Initialization Phase. We start the process by firing 1 spp path and photon rays. Path samples and photon samples are stored as point clouds in this particular phase to initialize our spatial grid; basically, we first build a regular grid given the spatial extent discovered by path samples, and then subdivide the grid given the initial per-voxel photons. Specifically, we use the collected path samples (after multiple bounces) to determine the bounding box of our spatial grid, which covers the *visible* part of the scene. We construct a regular grid by uniformly dividing the bounding box at a relatively coarse level (Figure 3.5). Next, for each voxel in this uniform grid, we iteratively sub-partition the voxel into a local binary tree through the median photon, which allows both sub-voxels to have a decent number of samples for reconstruction to start with, based on the number of photons that arrive at the voxel; this is also repetitively done in the following iterative update phase in a similar way but with a different splitting plane. Note that this initialization phase produces an initial hierarchical spatial grid – a coarse regular grid with local shallow KD-Trees. The initial grid is still coarse but relatively denser than a single shallow KD-Tree used in early stages in previous work [111]. This enables reconstructing more locally representative sampling maps, leading to better path guiding at early iterations in our framework, and better utilizes the benefits of our pre-trained deep neural network. The local KD-Trees are relatively shallow at this phase because photons are sparse, which are further subdivided in the iterative phase.

Iterative Update Phase. The spatial structure after the initialization phase can still be too coarse for a later time in rendering. Therefore, we iteratively subdivide it into a finer hierarchical grid. Our hierarchical grid is built to adapt to the complexity of incident light fields. We leverage the statistics of accumulated photons in each valid voxel for possible subdivisions. In particular, we consider M_j – the total number of photons hitting the valid voxel j. A voxel is split into two sub-voxels if $M_j > M_{\text{thr}}$, where M_{thr} is a predefined threshold. We recursively apply our subdivision criterion to sub-voxels. Unlike the initialization phase, we always use middle planes (instead of median) for splitting in this phase (same as [111]), since we do not store any photon point cloud anymore for the sake of memory, and this strategy is generally sufficient based on our observation. Once a voxel is subdivided, its two sub-voxels are reset as invalid waiting to be re-evaluated, inheriting the original noisy map with halved power values and accumulating photons from the subsequent iterations once marked valid again. This photon-based subdivision process allows these complex voxels to utilize more local and accurate sampling maps, thus leading to more efficient renderings.

Sampling Map Reconstruction. Apart from determining the subdivision in the hierarchical grid, the main goal of tracing the per-iteration photons is to reconstruct the per-voxel sampling maps for path guiding. For any valid voxel (detected by camera paths), we accumulate photon power to compute the noisy power map of the voxel, as expressed by Equation 3.10. The power map accumulates all hitting photons $\Delta\Phi_p$ in the voxel through the current and previous iterations, which gets normalized to a noisy sampling map S_t as the input to the neural reconstruction module (Section 3.3.5) in iteration t. Once sampling maps are reconstructed and updated in one iteration, they are used to guide the path sampling in the next iteration.

Path Guiding and One-Sample MIS. In any iteration, if a path hits a voxel that does not have a sampling map, we use standard BSDF sampling at the bounce point; the voxel is then marked as valid and starts accumulating photons immediately in the same iteration, enabling guiding in the subsequent iterations. On the other hand, once a path ray hits a valid voxel that has a reconstructed sampling map, path guiding can be achieved by importance sampling on the map (where CDF is built by GPUs). Since our sampling map only considers the incident radiance, we apply a one-sample MIS similar to previous works to combine guided sampling and BSDF sampling, as discussed in Equation 3.5. The combined sampling strategy however requires a parameter α that determines how often either sample strategy is selected. Usually, $\alpha = 0.5$ is a simple choice and performs reasonably well. An α that is learned via online optimization [110] is also presented for better performance. Here we propose a simpler alternative method to achieve a similar goal, which can also serve as a better initialization for those approaches that try to search for an optimal α.

We present a heuristic α computation technique based on collected path statistics; though simple, it results in effective per-voxel α_j in practice. In particular, we initialize $\alpha_j = 0.5$ in each valid voxel. Once a full path is constructed in rendering (either connect to or miss the light), we collect the reflected radiance contribution for every bounce point b on the path as $L_b = L(\boldsymbol{x}_b, \omega_b) \cos\theta_b f_r(\boldsymbol{x}, \omega_b, \omega_o)$. Meanwhile, for each voxel j, we accumulate all bounce contributions L_b (where $\boldsymbol{x}_b \in \Delta A_j$) in $L_{j,\text{BSDF}}$ and $L_{j,\text{Guide}}$, according to from which distribution ω_b is sampled. We also record the number of bounces sampled by two strategies as $Q_{j,\text{BSDF}}$ and $Q_{j,\text{Guide}}$. Once $Q_{j,\text{BSDF}} \geq Q_{\text{Thr}}$ and $Q_{j,\text{Guide}} \geq Q_{\text{Thr}}$ sub-paths have been collected in the voxel, we use the ratio of the averaged $L_{j,\text{BSDF}}$ and $L_{j,\text{Guide}}$ to update the mixing weight α_j:

$$\alpha_j = \frac{\overline{L}_{j,\text{BSDF}}}{\overline{L}_{j,\text{BSDF}} + \overline{L}_{j,\text{Guide}}}, \tag{3.14}$$

where $\overline{L}_{j,\text{BSDF}} = L_{j,\text{BSDF}}/Q_{j,\text{BSDF}}$ and $\overline{L}_{j,\text{Guide}} = L_{j,\text{Guide}}/Q_{j,\text{Guide}}$. Correspondingly, our one-sample MIS is expressed by:

$$p_{\text{MIS}}(\omega_i) = \frac{\overline{L}_{j,\text{BSDF}}}{\overline{L}_{j,\text{BSDF}} + \overline{L}_{j,\text{Guide}}} p_{\text{BSDF}}(\omega_i) + \frac{\overline{L}_{j,\text{Guide}}}{\overline{L}_{j,\text{BSDF}} + \overline{L}_{j,\text{Guide}}} p_{\text{guide}}(\omega_i). \tag{3.15}$$

We set $\alpha_j = 1$ if BSDF is a delta function and clamp α_j between 0.2 and 0.8 otherwise to handle statistical instability. This heuristic mixing weight considers the data that reflects the actual performance of BSDF sampling and guiding sampling, leading to effective mixed sampling in path guiding.

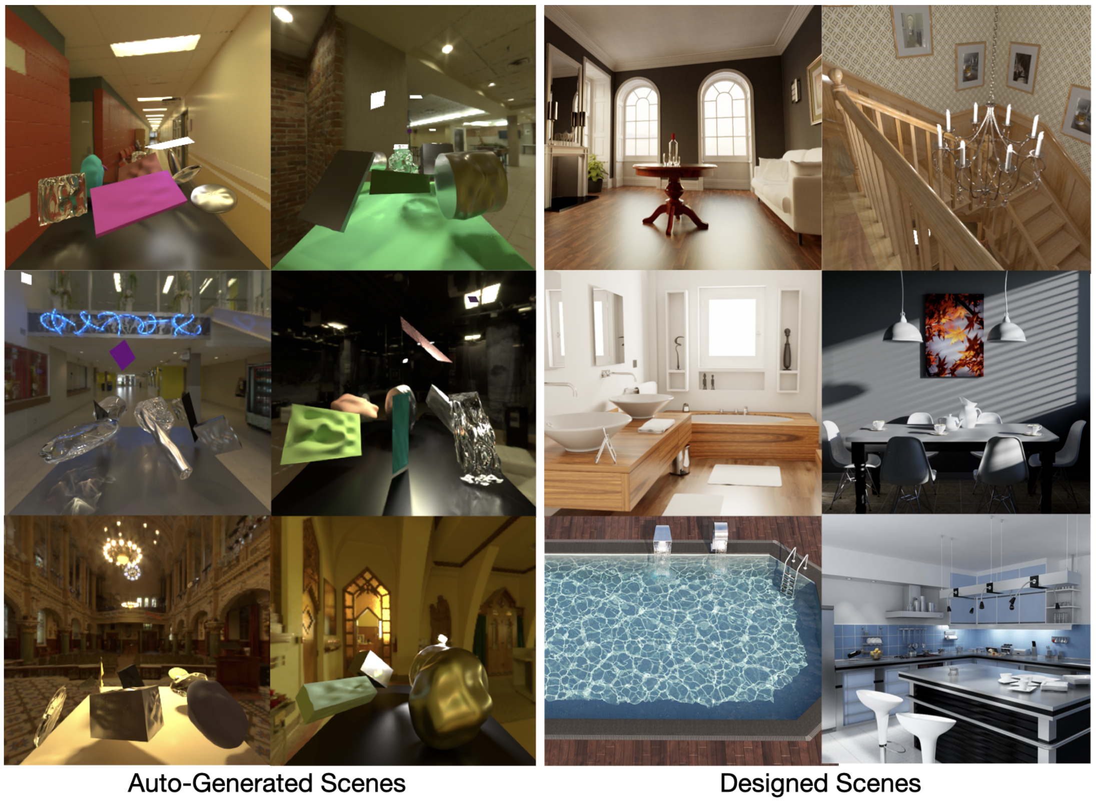


Figure 3.6. Example scenes used for training our proposed neural network.

Progressive Rendering. Our learning-based approach can reconstruct high-quality sampling maps from sparse photons in early iterations. We therefore leverage all path samples after the initialization phase for rendering the final image. While we can keep iteratively tracing more rays and refining the sampling maps, our reconstructions are often of sufficient quality after $T = 4{\sim}10$ iterations. Continuing tracing more photons afterward merely leads to marginal sampling improvement, and the extra overhead from running the neural network cannot pay off. Therefore, we choose to stop iterating after $T = 4{\sim}10$ depending on light transport complexity, fix the per-voxel sampling maps, and run a final path tracing pass (with N_f spp) guided by the latest maps.

3.3.7 Data Synthesis for Sampling Reconstruction

We create a large-scale dataset to train our sampling map reconstruction network. The dataset consists of both designed scenes and auto-generated scenes, as shown in Figure 3.6. We collect available scenes designed by researchers and artists from previous work and several websites [9, 74, 108, 164, 109, 107]. This leads to 28 designed scenes, including multiple realistic indoor and outdoor scenes; we use 20 from them in our training set and the rest for testing our algorithm. To enhance the generalizability of our neural network, we further enlarge our training set by procedurally generating 500 scenes using randomized shape primitives, materials, and area lights, similar to [212, 195]. We also leverage a complex lighting dataset [44] and randomly select an environment map for each generated scene as its additional illumination. This auto-generation process increases the diversity and complexity of our training scenes, leading to better generalization on testing scenes.

3.4 The Superior Model: Hierarchical Guiding from Hybrid Samples

Our first attempt (Section 3.3) along the direction of clever path sampling results into an initial publication [214]. Subsequently, we re-invented our method to throw light on resolving some of the previously encountered limitations.

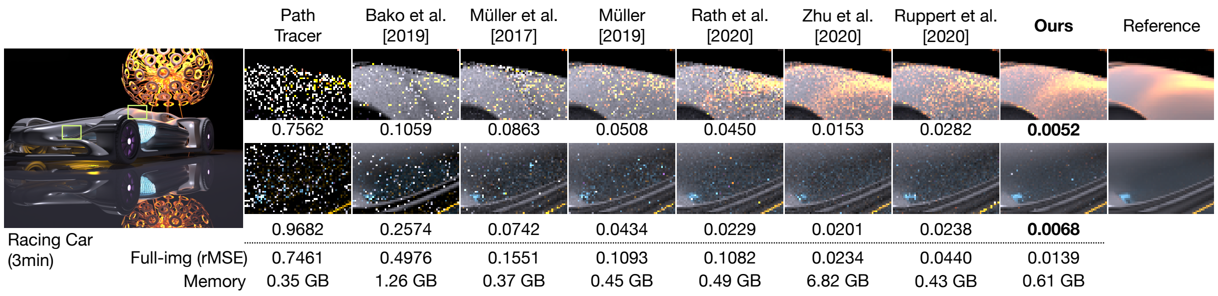
3.4.1 Overview of Advancement

Path guiding is a promising technique to reduce the variance of path tracing. Although existing online path guiding algorithms can eventually learn good sampling distributions given a large amount of time and samples, the speed of learning becomes a major bottleneck. In this section, we accelerate the learning of sampling distributions by training a lightweight neural network offline to reconstruct from sparse samples. Uniquely, we design our neural network to directly operate convolutions on a sparse quadtree, which regresses a high-quality hierarchical sampling distribution. Our approach can reconstruct reasonably accurate sampling distributions faster, allowing for efficient path guiding and rendering. In contrast to the recent offline neural path guiding techniques that reconstruct low-resolution 2D images for sampling, our novel hierarchical framework enables more fine-grained directional sampling with less memory usage, effectively advancing the practicality and efficiency of neural path guiding. In addition, we take advantage of hybrid bidirectional samples including both path samples and photons, as we have found this more robust to different light transport scenarios compared to using only one type of sample as in previous work. Experiments on diverse testing scenes demonstrate that our approach often improves rendering results with better visual quality and lower errors. Our framework can also provide the proper balance of speed, memory cost, and robustness.

3.4.2 Motivation and Methodology

The simple and flexible Monte-Carlo path tracing algorithm has become the gold standard for physically-based rendering. However, a major drawback is the slow convergence problem, leading to unpleasant Monte Carlo noise in the rendered image. In recent years, researchers have successfully tried many denoising and filtering techniques to reduce the noise level [15, 5, 172]. However, the denoised image is no longer unbiased and sometimes has remaining low-frequency artifacts.

Path guiding is a promising direction to reduce path tracing variance while remaining



	Path Tracer	Bako et al. [2019]	Müller et al. [2017]	Müller [2019]	Rath et al. [2020]	Zhu et al. [2020]	Ruppert et al. [2020]	**Ours**	Reference
	0.7562	0.1059	0.0863	0.0508	0.0450	0.0153	0.0282	**0.0052**	
	0.9682	0.2574	0.0742	0.0434	0.0229	0.0201	0.0238	**0.0068**	
Full-img (rMSE)	0.7461	0.4976	0.1551	0.1093	0.1082	0.0234	0.0440	0.0139	
Memory	0.35 GB	1.26 GB	0.37 GB	0.45 GB	0.49 GB	6.82 GB	0.43 GB	0.61 GB	

Figure 3.7. We present a hierarchical neural path guiding framework which uses both path and photon samples to reconstruct high-quality sampling distributions. This RACING CAR scene includes both complex direct and indirect illumination that are difficult for traditional path tracing to render. Traditional guiding methods [111, 110, 132] can reconstruct hierarchical sampling distributions (quadtrees) via online learning for multi-bounce path guiding. However, the online learning process is relatively slow, which results in noisy sampling maps for a long time, restricting the guiding efficiency. Bako et al. [6] leverages offline deep learning, but it can only guide the first bounce, which naturally cannot outperform traditional online methods for such a scene with strong global illumination. Ruppert et al. [138] introduces parallax compensation and uses mixture models (VMMs) to represent sampling distributions. However, the use of analytical mixtures limits the capability to represent complex radiance fields from sparse path samples, which requires careful strategies for merging and splitting of mixture components. The recent photon-driven work [214] can support multiple bounces using an offline-trained network, producing better renderings than many previous methods. However, this method uses standard regular 2D images (unlike quadtrees) to represent lighting distributions, requiring the largest memory consumption, limiting its scalability to large-scale scenes. Our approach enables neural reconstruction of the traditional hierarchical representation via an offline-trained novel network; we can effectively reconstruct accurate quadtree-based sampling distributions, consuming less system memory than [214]. Our approach also combines both path and photon samples, which is more robust against different light-transport scenarios. As a result, we can achieve better quantitative (reflected by lower rMSE–relative Mean Squared Error) and qualitative results, with moderate memory cost comparable to traditional online methods that do not use deep neural networks.

unbiased. The key idea is to learn a better sampling distribution (approximating the incident light field or some variant of it) at arbitrary scene locations and guide camera rays towards the light source. Previous methods [111, 132, 138] often require a slow online learning process to obtain accurate sampling distributions for path guiding. While some recent works [6, 214] use offline-trained neural networks, their methods require large system memory and can only reconstruct sampling distributions at a low resolution, restricting the accuracy and efficiency of path guiding.

In this work, we present a novel neural path guiding approach that can effectively reconstruct accurate hierarchical high-resolution sampling distributions, leading to efficient path guiding and rendering. Our approach uses an offline-trained neural network to accelerate online learning in traditional path guiding. Unlike previous offline neural methods that represent a distribution using a uniform grid (as a 2D image), we consider the classical quadtree-based representation, allowing for efficient high-resolution distribution modeling. As shown in Figure 3.7, our approach successfully advances the efficiency of neural path guiding, leading to better rendering quality with moderate memory costs.

We present a novel deep neural network for efficient hierarchical distribution reconstruction. Our technique is inspired by the octree networks [181] in 3D geometry processing. We propose to operate deep 2D convolutions directly on a sparse quadtree that represents a 2D angular sampling distribution, enabling an efficient hierarchical reconstruction. Our network can adaptively adjust the tree structure in reconstruction, which learns the proper angular resolution for each sampling solid angle bin. This results in high-quality distributions that accurately express the incident light fields. In contrast to the standard convolutional neural networks (CNNs) that can only regress low-resolution sampling maps [214], our network hierarchically regresses a compact quadtree that represents the same distribution at a much higher resolution using less memory. The adaptivity and compactness of our hierarchical reconstruction improve the scalability to large-scale complex scenes where a large number of sampling distributions need to be stored on numerous mesh surfaces.

Previous path guiding work uses either path samples [111, 110, 132, 138] or photons [76, 175, 176] to reconstruct an incident radiance field which is then converted to a sampling distribution at any scene location. Our hierarchical neural reconstruction can potentially support either input sample independently. However, path samples and photons can perform differently depending on the actual light transport cases (see extreme examples in Figure 3.9). When the scene contains caustics produced by transparent objects or tiny light sources, photons are more efficient since it is difficult for path samples to quickly find a valid direction towards the light. On the other hand, path samples are a better choice when some light sources do not illuminate the visible regions of the scene, since many photons can be invisible and useless in this case. In this work, we use both of them and let the neural network figure out how to effectively combine the hybrid samples into a single output sampling distribution. Therefore, our approach is more robust to general scenes with unknown light transport scenarios.

In summary, our main contributions are:

- We propose a novel learning-based framework that can reconstruct a *hierarchical* sampling distribution from sparse samples with a moderate memory cost;
- We consider *hybrid* input samples including both path samples and photons for path guiding, leading to higher robustness and generality on diverse light transport cases.

3.4.3 Importance Sampling Revisit

Rendering equation. To render a scene using light transport simulation, our goal is to solve the rendering equation [81]:

$$L(\boldsymbol{x},\omega_o) = L_e(\boldsymbol{x},\omega_o) + \int_\Omega L_i(\boldsymbol{x},\omega_i) f_r(\boldsymbol{x},\omega_i,\omega_o)\cos\theta_i d\omega_i, \tag{3.16}$$

where the outgoing radiance $L(\boldsymbol{x},\omega_o)$ in direction ω_o at each surface point $\boldsymbol{x}$ equals the sum of the surface emission $L_e(\boldsymbol{x},\omega_o)$ and the reflection from the incoming radiance $L_i(\boldsymbol{x},\omega_i)$ of every direction ω_i that has angle θ_i to the surface normal over the hemisphere Ω. The Bidirectional

Scattering Distribution Function (BSDF) $f_r(\boldsymbol{x}, \omega_i, \omega_o)$ describes how much radiance can be scattered to ω_o from ω_i.

The integration $L_r(\boldsymbol{x}, \omega_o) = \int_\Omega L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i d\omega_i$ in Equation 3.16 is computed by Monte Carlo (MC) estimation [169] in the path tracing algorithm:

$$L_r(\boldsymbol{x}, \omega_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)} \tag{3.17}$$

where N is number of samples and $p(\omega_i)$ is the probability density function (PDF) of sampling direction ω_i (i.e., importance sampling). When N is sufficiently large, the variance of $L_r(\boldsymbol{x}, \omega_o)$ reduces, and path tracing gradually converges to the noise-free result.

In many challenging light transport scenarios, the convergence is very slow, which is the major drawback of Monte Carlo path tracing. Fortunately, we can greatly speed up the variance reduction by sampling from a better PDF $p(\omega_i)$ that resembles the integrand $L_i(\boldsymbol{x}, \omega_i) f_r(\boldsymbol{x}, \omega_i, \omega_o) \cos\theta_i$. However, the incident radiance field $L_i(\boldsymbol{x}, \omega_i)$ is unknown in the beginning, so standard path tracing only leverages the BSDF for importance sampling:

$$p_{\text{BSDF}}(\omega_i) \propto f_r(\boldsymbol{x}, \omega_i, \omega_o) \tag{3.18}$$

Guiding using path samples. In contrast, path guiding is a method to evaluate the incident light $L_i(\boldsymbol{x}, \omega_i)$ and set the PDF to be proportional to some terms related to it. Many previous papers [111, 110] use early (or extra) Monte Carlo path samples to compute a sampling distribution as:

$$p_{\text{guide}}(\omega_i) \propto L_i(\boldsymbol{x}, \omega_i) \cos\theta_i, \tag{3.19}$$

which expresses the incident light field (the cosine term is sometimes associated to the BSDF sampling in Equation 3.18). In practice, this guided sampling is often combined with BSDF sampling, using one-sample Multiple Importance Sampling (MIS) [171]:

$$p(\omega_i) = \alpha p_{\text{BSDF}}(\omega_i) + (1-\alpha) p_{\text{guide}}(\omega_i) \tag{3.20}$$

where the coefficient α determines the chance of selecting BSDF over guiding for importance sampling.

However, since $L_i(\boldsymbol{x}, \omega_i)$ is also from the noisy Monte Carlo samples [111], the estimates are also noisy and can have high variance, making the sampling inefficient. Recently, Rath et al. [132] introduce a variance-aware guiding technique, leveraging a new target sampling function that considers the variance:

$$p_{\text{guide-var}}(\omega_i) \propto \sqrt{\mathbb{E}[L_i^2(\boldsymbol{x}, \omega_i)] \cos^2 \theta_i} \tag{3.21}$$

where $\mathbb{E}[\cdot]$ represents the expectation. Additionally, they also take the surface material into account, resulting in the BSDF marginalized product sampling [132] used for path guiding:

$$p_{\text{guide-var-prod}}(\omega_i) \propto \sqrt{\mathbb{E}_{\omega_o}[f_r^2(\boldsymbol{x}, \omega_i, \omega_o) \mathbb{E}[L_i^2(\boldsymbol{x}, \omega_i)] \cos^2 \theta_i]} \tag{3.22}$$

Our framework generally supports various sampling functions. We take advantage of the advanced variance-aware technique (Equation 3.22) to generate our results by default, leading to better quality than our results with the traditional distribution (Equation 3.19).

Guiding using photons. In some special light transport cases such as caustics from transparent objects or tiny light sources that are hard to find through Monte Carlo sampling, most path samples are terminated before reaching any light, leading to more noisy $L_i(\boldsymbol{x}, \omega_i)$ estimation. Compared to path samples, photons are often a better choice in these scenarios, which have been used for path guiding by previous work [76, 175, 214]. Each photon p carries a small portion of the emitter power (radiant flux) $\Delta\Phi_p$ and its direction ω_p indicates where the light comes from. The power $\Phi(\boldsymbol{x}, \Delta\Omega)$ that flows through a solid angle footprint $\Delta\Omega$ in local surface area A

is computed via integrating the incident radiance $L_i(\boldsymbol{x}, \omega_i)$ where $\omega_i \in \Delta\Omega$:

$$\Phi(\boldsymbol{x}, \Delta\Omega) = \int_A \int_{\Delta\Omega} L_i(\boldsymbol{x}, \omega_i) \cos\theta_i d\omega_i d\boldsymbol{x}. \tag{3.23}$$

The target distribution can be expressed as [214]:

$$p_{\text{guide-photon}}(\omega_i) \propto \Phi(\boldsymbol{x}, \Delta\Omega)/\Delta\Omega = \sum_{\omega_p \in \Delta\Omega, \boldsymbol{x} \in A} \Delta\Phi_p/\Delta\Omega \tag{3.24}$$

This sampling distribution similarly approximates Equation 3.19, but it is evaluated by the summation of the surrounding photon power, instead of the Monte Carlo estimation of path samples.

In practice, it is hard to know whether a path sample or photon is better for an unknown scene; two extreme examples are shown in Figure 3.9. Therefore, in this work, we choose to use both of them (i.e., hybrid samples), although our framework also directly applies to a single type of sample. Combining path samples and photons is challenging since they distribute very differently, and there is no obvious and cheap way to combine them through re-weighting (VCM [46]. One concurrent work [63] design specific techniques to address a similar issue in radiance estimation, which however cannot be easily extended to distribution estimation). Therefore, we use a neural network that learns to combine their values and reconstructs a single sampling distribution (Section 3.4.6) that is then used for path guiding.

3.4.4 Path Guiding Pipeline with Tree Structure

The entire framework is illustrated in Figure 3.8. We trace both photon and path samples (i.e., hybrid samples), and deposit them into a local quadtree representation of the sampling distribution stored in a local spatial caching node as shown in Figure 3.8(a) and (b). This step is similar to the online quadtree construction in [111]; it leads to noisy distributions unless a large number of samples are deposited. We instead propose to use a deep neural network (pre-trained)

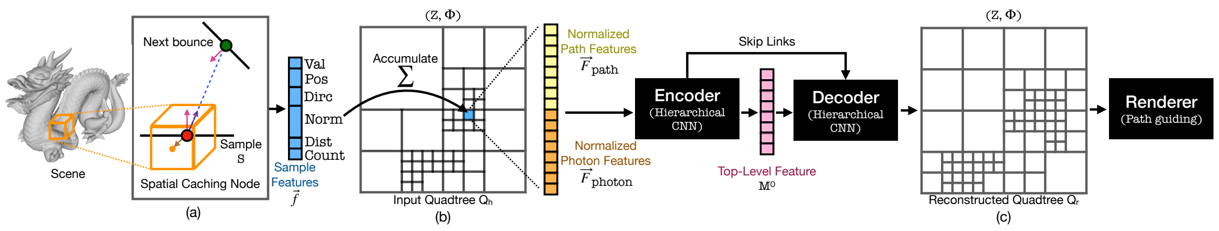

Next bounce
Sample s
Scene
Spatial Caching Node
(a)
Val
Pos
Dirc
Norm
Dist
Count
Sample Features $\vec{f}$
(z, Φ)
Accumulate $\sum$
Input Quadtree Q_h
(b)
Normalized Path Features $\vec{F}_{path}$
Normalized Photon Features $\vec{F}_{photon}$
Skip Links
Encoder (Hierarchical CNN)
Decoder (Hierarchical CNN)
Top-Level Feature M^0
(z, Φ)
Reconstructed Quadtree Q_r
(c)
Renderer (Path guiding)

Figure 3.8. High-level illustration of the proposed neural path guiding framework. The scene is partitioned into many spatial caching nodes (voxels). Each voxel collects all the samples that arrive at it (a) and uses the sample information to adatpively construct a quadtree Q_h (b), parameterized in the cylindrical coordinates ω_i to (z, ϕ). Each sample contains its value along with some auxiliary features including the relative position $\vec{p}$, direction ω_i, distance $\vec{d}$, normal $\vec{n}$, and sample count $c = 1$, which leads to a feature vector $\vec{f}$ that will be accumulated into a leaf l of the quadtree Q_h (Section 3.4.6). The accumulation is applied separately to path samples and photons, resulting in two independent feature vectors $\vec{\mathbb{F}}_{\text{path}}$ and $\vec{\mathbb{F}}_{\text{photon}}$ (Equation 3.26). We propose a novel neural network that can directly operate convolutions on quadtree distributions (Section 3.4.6), which has an architecture with hierarchical encoder and decoder. We train our network offline that learns to hierarchically regress accurate sampling distributions from noisy inputs. The pre-trained network can reconstruct a high-quality quadtree Q_r (c) from the input Q_h; the reconstructed Q_r is stored in the spatial voxel and later used for path guiding (Section 3.4.7).

to hierarchically reconstruct accurate quadtree distributions from the noisy ones in Figure 3.8(c). In the following sections, we first describe the steps of building an initial quadtree at arbitrary scene locations and depositing hybrid samples into it (Section 3.4.5). Next, Section 3.4.6 presents the key component of our framework: a novel neural network to reconstruct high-quality hierarchical sampling distributions using both the initial noisy path and photon distributions as input. Finally, we discuss the details of adaptively caching the reconstructed distributions at different locations in the scene and rendering of the final image (Section 3.4.7). Thereafter, Section 3.6 provides the implementation details of neural training, sample tracing, and rendering. Experiments on diverse testing scenes in Section 3.7 justify the effectiveness of our proposed framework.

3.4.5 Hierarchical Distribution Representation from Hybrid Samples

As we discussed in Section 3.4.3, we need to collect path samples and/or photons to learn a directional sampling distribution that resembles the incident radiance field at arbitrary scene

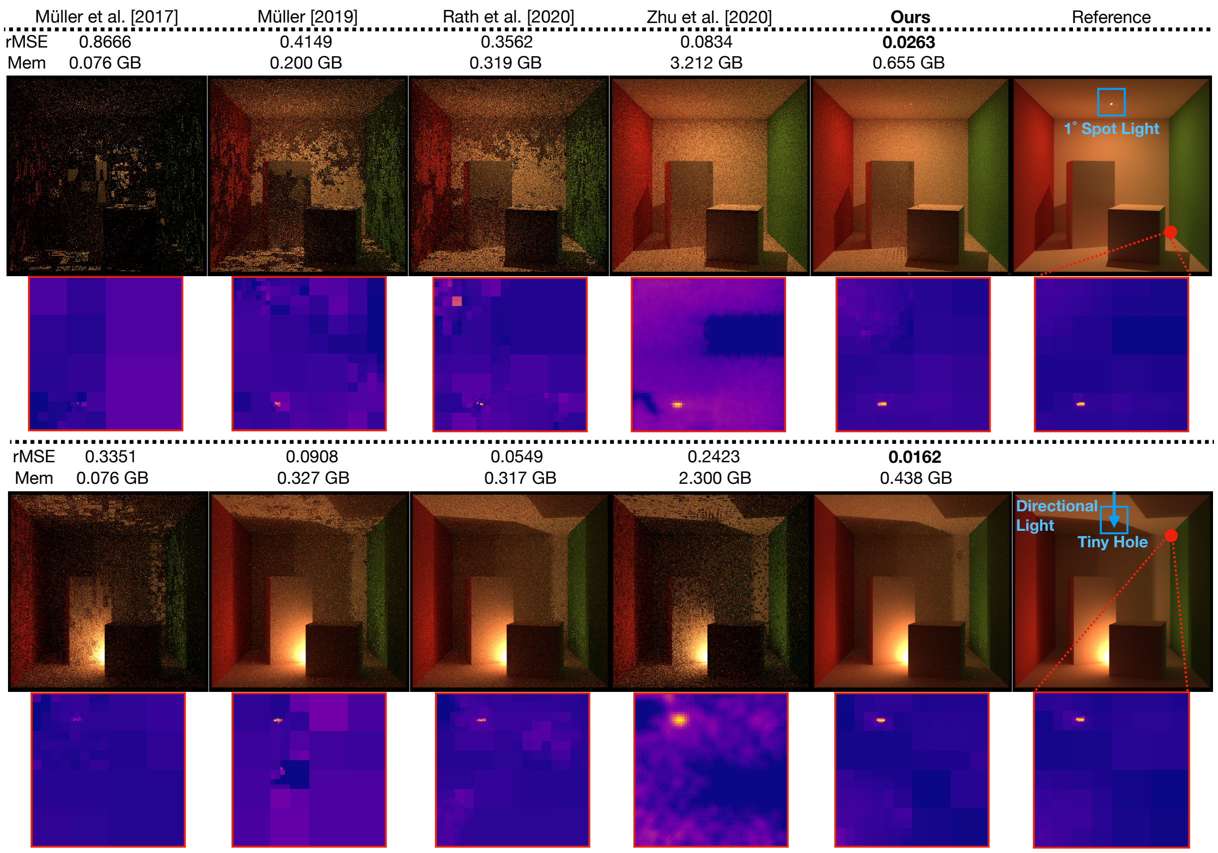

Müller et al. [2017]
Müller [2019]
Rath et al. [2020]
Zhu et al. [2020]
Ours
Reference
rMSE 0.8666 0.4149 0.3562 0.0834 0.0263
Mem 0.076 GB 0.200 GB 0.319 GB 3.212 GB 0.655 GB
1° Spot Light
rMSE 0.3351 0.0908 0.0549 0.2423 0.0162
Mem 0.076 GB 0.327 GB 0.317 GB 2.300 GB 0.438 GB
Directional Light
Tiny Hole

Figure 3.9. Extreme conditions. We compare our method with previous path guiding methods, running with equal time, on two Cornell Box scenes that have two different extreme light transport settings. We turn on the next event estimation for all methods in this experiment. Previous methods utilize either path samples [111, 110, 132] or photons [214] as input for path guiding, which cannot work well on both cases at the same time. In the first row, the scene is illuminated by a very small 1° *spotlight* (facing upwards) located very close to the roof. This setting is extremely hard for path-based methods [111, 110, 132] since the light is hard to connect to; yet, the photon-based method [214] still works well. On the other hand, the second row shows a scene illuminated by a *directional light* coming from the top, while the roof only has a very tiny hole that can receive this light. While path-based methods can still be effective for this setting, the photon-based method [214] cannot work well (even empowered by deep learning) since most photons will be blocked by the roof and not useful at all. Our novel neural approach leverages both path and photon samples as input, and can successfully work on both challenging cases. We also show the corresponding sampling distributions reconstructed by all methods. Note that these methods may have different ground-truth target sampling distributions (see Section 3.4.3). We only show our ground truth (the target of [132]) as the reference. While the target sampling functions are different, we can still observe that our neurally reconstructed quadtrees are of higher quality than the noisy quadtrees reconstructed traditionally by [111, 110, 132]; ours also contain sharper details than the regular image representation of [214]. Our quadtrees are reasonably accurate compared to the reference.

Table 3.2. List of notations used in Section 3.4.5, Section 3.4.6, and Section 3.4.7.

	Notation	Meaning
Hierarchical input structure (Section 3.4.5)	$\mathbb{S}$	Sample
	l	Quadtree leaf
	ω_i	Sample direction
	$(\mathbb{Z}, \Phi)$	Directional sampling space
	$V_{\mathbb{S}}$	Sample value
	$\mathbb{A}$	Accumulated sample value
	$\mathbb{Q}_h$	input quadtree from online accumulation
Neural network framework (Section 3.4.6)	$\mathbb{Q}_r$	Reconstructed quadtree from the neural network
	$\mathbb{Q}_{\text{gt}}$	Target (groundtruth) quadtree
	$\vec{f}$	Per-sample feature vector
	$\vec{\mathbb{F}}$	Per-leaf feature vector
	F_{conv}	Convolution result
	$\mathbb{M}$	Per-level feature map
	$\mathbb{V}$	Predicted relative value to the parent node
	p_{leaf}	Predicted probability of node being a leaf
	m, n	Encoding and decoding tree level
	q, q_c	Decoded tree node and one of its children
	$\mathbb{L}_{Q_r}$	Loss function
	$\mathcal{P}$	Pooling
	$\mathcal{S}$	Convolution
	$\mathcal{U}$	Upsampling
	$\mathcal{T}$	Node type classifier
	$\mathcal{R}$	Value regressor
Sampling and rendering (Section 3.4.7)	r_{init}	Initial grid resolution
	$\mathbb{G}$	Adaptive hierarchical hash grid
	$\mathbb{B}_{\text{spt}}$	KD-tree in each voxel
	k_{spt}	Spatial subdivision threshold
	t, $\mathbb{T}$	Current and total iteration(s)
	α	One-sample MIS coefficient

locations. Compared to the previous neural path guiding work [6, 214], we hierarchically build a quadtree instead of a uniform 2D grid (image) to represent the distribution. Hybrid samples are traced and stored in the tree, which are later provided to our hierarchical neural network for sampling distribution reconstruction (Section 3.4.6).

Our quadtree-based distributions are stored in small spatial caching nodes distributed within the scene, as shown in Figure 3.8(a). Later in Section 3.4.7, we discuss the details of adaptively partitioning the scene space into local regions of different sizes for efficient spatial caching. We keep two quadtrees in each spatial node: one records the online traced hybrid samples, representing a noisy distribution and used as network input; the other is the output of the network, representing an accurate sampling distribution for path guiding. The initial noisy quadtree collects local samples that arrive at the node, containing rich information of the local incident radiance field.

Quadtree Representation. We use the 2D cylindrical coordinates to parameterize the angular space; each unit vector (x, y, z) is mapped to (z, ϕ), where $\phi = \arctan(y/x)$. A quadtree $\mathbb{Q}_h$ is built to hierarchically cover the space of $(z \in \mathbb{Z}, \phi \in \Phi)$ at each spatial node, recording the hybrid samples traced at rendering time (Figure 3.8(b)).

Accumulating Hybrid Samples. Once a sample $\mathbb{S}$ (either path or photon), carrying a sample quantity $V_{\mathbb{S}}$, arrives at a particular spatial node, we convert its incident direction $\omega_i = (x, y, z)$ to the cylindrical space mentioned above, and deposit it to a corresponding leaf node l of the quadtree $\mathbb{Q}_h$. In particular, we leverage a stochastic box filter [110], which deposits the sample value $V_{\mathbb{S}}$ into a single neighboring tree leaf l around its original direction ω_i; this is equivalent to splatting the sample with a box filter into the quadtree.

Since path samples and photons have different radiometric units (Section 3.4.3), we keep two separate accumulators $\mathbb{A}^l_{\text{path}}$ and $\mathbb{A}^l_{\text{photon}}$:

$$\begin{aligned} \mathbb{A}^l_{\text{path}} &= \sum V^l_{\mathbb{S}_{\text{path}}} \\ \mathbb{A}^l_{\text{photon}} &= \sum V^l_{\mathbb{S}_{\text{photon}}} \end{aligned} \tag{3.25}$$

where $V^l_{\mathbb{S}_{\text{path}}}$ and $V^l_{\mathbb{S}_{\text{photon}}}$ are splatted sample quantities in leaf l.

Quadtree Subdivision. Initially, the tree $\mathbb{Q}_h$ has a single node. To effectively construct $\mathbb{Q}_h$ as a hierarchical structure, we iteratively trace samples (Section 3.4.7) and subdivide the tree accordingly. Specifically, $\mathbb{Q}_h$ is adaptively refined after the samples in the current iteration are deposited based on a criterion [111]: if a node value $\mathbb{A}^l_{\text{path}}$ or $\mathbb{A}^l_{\text{photon}}$ is greater than $k\%$ (we empirically find that $0.5\% \sim 1\%$ is a reasonable threshold) of its total value ($\sum_l \mathbb{A}^l_{\text{path}}$ or $\sum_l \mathbb{A}^l_{\text{photon}}$) in $\mathbb{Q}_h$, the node is split into four equal-sized child nodes where each of them is assigned $1/4$ of the parent value, otherwise it remains as a leaf node. This criterion is applied recursively to each node in the tree. After $\mathbb{Q}_h$ is updated, it is used to collect future samples in the next iteration, so that $\mathbb{Q}_h$ can be repeatedly refined to better versions. This strategy allows $\mathbb{Q}_h$ to have higher directional resolution when the radiance of an incident direction is large. Note that if only path samples are considered (as in previous work [111]), then only $\mathbb{A}^l_{\text{path}}$ is used to build and refine $\mathbb{Q}_h$.

This iteratively-refined quadtree $\mathbb{Q}_h$ can in fact model an accurate sampling distribution when the number of accumulated samples is large enough. However, this requires a large number of iterations and a long time for accumulation, which cannot promptly provide reliable sampling distributions. Especially at the beginning of rendering, the accumulated sampling quadtrees are highly noisy and inadequate for path guiding. In Section 3.4.6, we design a novel neural network to handle the hybrid input samples stored in each leaf l.

3.4.6 Neural Refinement of Quadtrees

In this section, we introduce our novel hierarchical neural network that can effectively convert the deposited hybrid samples (Section 3.4.5) to a high-quality sampling distribution for path guiding. We first discuss the motivation of applying neural networks in the context of sampling distributions. Next, we present our network input, the convolutional module applied on a quadtree, and the detailed neural architecture. Finally, we introduce our loss function to train the neural network.

Motivation of Neural Reconstruction Framework. As discussed in Section 3.4.5, directly reconstructing an accurate quadtree distribution $\mathbb{Q}_h$ via online accumulation usually requires a long time to trace a large number of samples, leading to low quality of sampling at early rendering times (as appears in previous work [76, 111]). We therefore seek to directly reconstruct an accurate quadtree distribution from the initial noisy quadtree; this can be seen as a traditional image reconstruction task (like denoising, inpainting, or restoration) in the $(\mathbb{Z},\Phi)$ space, except that now the task is applied on hierarchical trees instead of regular 2D images. Therefore, the standard CNN on a 2D grid image (e.g., [6, 214]) is no longer applicable, and we aim to design a new neural architecture that extends CNNs to hierarchical inputs and outputs. Meanwhile, prior works have been addressing a similar task in 3D geometry processing. They apply CNNs on octrees [181, 182] and hierarchical MLPs on grammar trees [96, 97] to achieve highly efficient 3D learning. We extend these 3D learning techniques to the reconstruction of sampling distributions, and we propose to apply neural convolutional operations on the 2D sampling quadtrees. Note that our neural framework can not only denoise the values of the input $\mathbb{Q}_h$, but also create an entirely separate hierarchical structure $\mathbb{Q}_r$ that can be different from $\mathbb{Q}_h$, better representing the target distribution. Our hierarchical neural reconstruction leverages the sparsity of the sampling distribution, processing and modeling directly on quadtrees; this allows for high-resolution modeling using low memory, which is not achievable when using regular images with CNNs. In addition, we also design our network to be compact enough for high computational and memory efficiency; this is ideal for path guiding, since it needs to simultaneously reconstruct sampling distributions at many different scene locations without introducing too much overhead to the rendering algorithm.

Input Hybrid Samples. As described in Section 3.4.5, when a new path sample or photon arrives, we convert its ω_i into (z,ϕ) and search $\mathbb{Q}_h$ to find its corresponding leaf node. Two separate value accumulators ($\mathbb{A}_{\text{path}}$ and $\mathbb{A}_{\text{photon}}$ in Equation 3.25) are used for adaptively refining $\mathbb{Q}_h$. However, using only one-channel sample values is insufficient for reconstructing a better quadtree. In this work, we collect additional auxiliary per-sample information and form a hybrid multi-channel feature vector, as illustrated in Figure 3.8(a). Specifically, each sample contains value $\mathbb{V}$ and additional features which include the local sample position $\vec{p}$, the sample direction ω_i, the distance d to the next bounce, the surface normal directions $\vec{n}$ of the current and next bounce, and the sample count $c = 1$. For path samples, we also append the BSDF value f_r to the vector. Finally, as shown in Figure 3.8(b), sample features $\vec{f}_{\text{path}}$ and $\vec{f}_{\text{photon}}$ are accumulated on each leaf l at tree level m, and then concatenated into a single feature vector $\vec{\mathbb{F}}^{m,l}$:

$$\begin{aligned}
&\vec{f}_{\text{path}} = (\mathbb{V},\omega_i,\vec{p},d,\vec{n},c,f_r)\ \vec{f}_{\text{photon}} = (\mathbb{V},\omega_i,\vec{p},d,\vec{n},c)\\
&\vec{\mathbb{F}}^{\text{acc}} = (\textstyle\sum\vec{f}_{\text{path}},\sum\vec{f}_{\text{photon}})\\
&\vec{\mathbb{F}}^{m,l} = \left(\frac{\vec{\mathbb{F}}^{\text{acc}}_{\text{path}}}{\max\limits_{Q_h}\vec{\mathbb{F}}^{\text{acc}}_{\text{path}}},\frac{\vec{\mathbb{F}}^{\text{acc}}_{\text{photon}}}{\max\limits_{Q_h}\vec{\mathbb{F}}^{\text{acc}}_{\text{photon}}}\right)
\end{aligned} \tag{3.26}$$

where $(,)$ means vector concatenation, and summations are computed for each leaf. $\max\limits_{Q_h}\vec{\mathbb{F}}^{\text{acc}}$ is the feature-wise maximum value within the entire quadtree $\mathbb{Q}_h$ after the summation, which is used for separately normalizing the input of path samples and photons. This normalization effectively removes the radiometric unit difference between path samples and photons. The sample direction ω_i is also implicitly included in the (z,ϕ) coordinates of $\mathbb{S}$.

Convolution on Quadtree. We propose to directly apply convolutions on the quadtree to process and regress the hierarchical feature data. In general, given a leaf l on level m in $\mathbb{Q}_h$, a convolutional layer outputs a new feature $F_{\text{conv}}^{m,l}$ via a linear operation that is applied on its neighbors (empty neighbor nodes are regarded as zeros) on the same tree level m:

$$\begin{aligned} F_{\text{conv}}^{m,l}[g] &= \sum_{c}\sum_{i}\sum_{j} W_{i,j,c}[g] \cdot \vec{\mathbb{F}}_{i,j,c}^{m,l} \\ \mathbb{M}^{m}[g][l] &= F_{\text{conv}}^{m,l}[g] \end{aligned} \tag{3.27}$$

where i and j are 2D indices of the neighbors inside the convolutional kernel W, c represents the channel index of input features, and g is the index of kernels (also the channel index of the output feature). Here $\mathbb{M}^m$ is a sparse 2D feature map containing the output features of all valid leaves. Note that this convolution on a quadtree (Equation 3.27) is not much different from the standard convolutional layer on a 2D image. However, each $\vec{\mathbb{F}}_{i,j,c}^{m,l}$ represents a feature in a quadtree leaf node instead of a standard pixel; unlike an image, leaf nodes on a single tree level m can distribute very sparsely, where only a few leaves contain actual features that require convolutions.

Moreover, accessing a neighbor within the convolutional kernel requires searching in the quadtree $\mathbb{Q}_h$ to get its stored features $\vec{\mathbb{F}}^{m,l}$ (Equation 3.26). This is non-trivial and can be much slower than the standard CNN on a regular image where any element in an array is immediately accessible. Fortunately, this neighboring search problem has been addressed by the 3D shape processing community using a faster hash table implementation [55, 181, 182] with an optimization on reducing hash table lookup times. In this work, we apply the same technique to speed up our CNN on quadtrees, enabling efficient quadtree convolutional operations. The same neighboring search is also naturally applied to pooling layers in our network. Note that, because of the sparsity of a quadtree, the network layers are applied only to the sparse nodes in each tree level m, which actually reduces the amount of computation compared to the standard dense CNNs.

Hierarchical Architecture. Our proposed neural architecture (Figure 3.10) contains a hierarchical encoder and decoder, with the skip links in between (Figure 3.8(c)). Each hierarchical

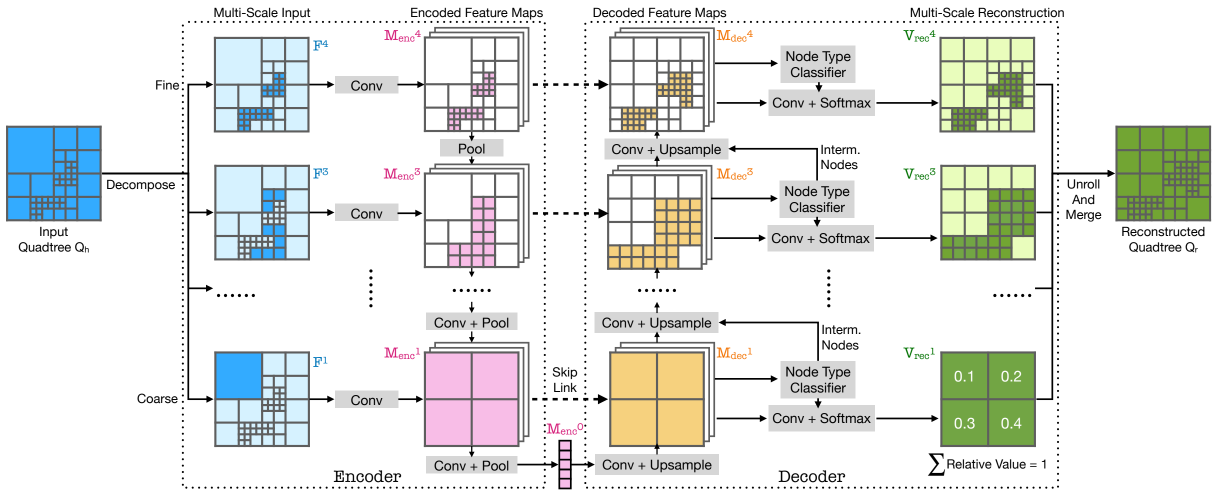


Figure 3.10. Our proposed hierarchical encoder-decoder architecture for reconstructing an accurate quadtree representation of sampling distributions. Here, we show an example containing only 4 levels. In practice, the input and output tree can have different levels ranging from 1 to 20. First, on each level m of the noisy input quadtree Q_h, we apply a series of convolutional and pooling layers to encode the sample features $\mathbb{F}^m$ to a neural feature map $\mathbb{M}_{\text{enc}}^m$. By repeatedly applying these operations hierarchically from the bottom level to the root node, we eventually encode and compress the whole Q_h into a single feature vector $\mathbb{M}_{\text{enc}}^0$. The decoder can be seen as the reverse of the encoder, which includes a series of convolutional and upsampling layers to extract new features $\mathbb{M}_{\text{dec}}^n$ and reconstruct a new quadtree Q_r that has new tree structure and values. On each decoding level n, we use convolutions followed by a SoftMax operation to regress a relative value $\mathbb{V}_{\text{rec}}^{n,q}$ for each node q with respect to its parent node value (therefore the summation of every four child nodes satisfy $\sum_q \mathbb{V}_{\text{rec}}^{n,q} = 1$). Meanwhile, an MLP classifier $\mathcal{T}_n$ predicts the type of each decoded node on that level, and sends all the intermediate nodes into the $(n+1)$-th level for further processing. Finally, the predicted values $\mathbb{V}_{\text{rec}}^{n,q}$ are converted and merged into the output quadtree Q_r. Note that encoding and decoding operations are applied only to the sparse nodes on each level, which is more computationally efficient compared to the standard CNNs that operate on dense image grids.

processing layer represents a corresponding tree level in the input $\mathbb{Q}_h$ or the output $\mathbb{Q}_r$.

Neural Hierarchical Quadtree Encoder. We take $\mathbb{Q}_h$ as the input and process the leaves from the bottom (finest) level to the top level. On each level m, we apply a series $\mathcal{S}_m$ of convolutions (Equation 3.27) and nonlinear ReLU activation functions on the accumulated feature vectors $\vec{\mathbb{F}}^{m,l}$. The output feature map $\mathbb{M}^m$ (Equation 3.27) is downsampled to the $(m-1)$-th quadtree level after the 2×2 average pooling $\mathcal{P}_m$, and is then fused with the feature map

$\mathbb{M}^{m-1}$ at the $(m-1)$-th level. This iterative encoding can be expressed as:

$$\mathbb{M}_{\text{enc}}^{m-1} = (\mathcal{P}_m(\mathbb{M}^m), \mathbb{M}^{m-1}) \tag{3.28}$$

where $\mathbb{M}_{\text{enc}}^{m}$ is the fused feature at level m. In summary, we start with the bottom tree level m_{max} in $\mathbb{Q}_h$ and combine the features from every coarser level until reaching the 0-th level (tree root).

Neural Hierarchical Quadtree Decoder. Our goal is to reconstruct a tree $\mathbb{Q}_r$, which can better represent the target sampling distribution from hybrid sample inputs. To do so, we design our decoder not only to regress the output distribution values at each tree level but also to determine if every node needs to be a leaf node or requires further subdivision. This allows the decoder to simultaneously build a new tree structure and reconstruct (denoise) leaf values.

The entire decoder can be seen as an inverse process of the encoder. After the multi-scale features $\mathbb{M}_{\text{enc}}^{m}$ are hierarchically extracted from $\mathbb{Q}_h$ via the encoder (Equation 3.28), we apply a series $\mathcal{S}_n$ of convolutions and ReLU activations to compute the feature $\mathbb{M}_{\text{dec}}^{n}$ at each decoding level n ($n = m = 0$ is the tree root). A 2×2 upsampling layer $\mathcal{U}_n$ on level n is also applied, subdividing a node into four equal-sized children, which reverses the operation of average pooling $\mathcal{P}_m$ in the encoder when $m = n$.

In order to obtain the final outputs, we apply final layers $\mathcal{S}_n$ to regress the distribution values and $\mathcal{T}_n$ to classify node types. In particular, $\mathcal{T}_n$ on level n predicts the type of each decoded node q, outputting the probability $p_{\text{leaf}}^{n,q}$ of the node being a leaf node. During inference, when $p_{\text{leaf}}^{n,q} > 0.5$ then the node is decoded as a leaf node, otherwise ($p_{\text{leaf}}^{n,q} < 0.5$) the current node is split into four children nodes in the next tree level. $\mathcal{R}_n$ is applied to regress a *relative* distribution value $\mathbb{V}_{\text{rec}}^{n,q}$ for each node q to its parent node at each level n; we apply the SoftMax in $\mathcal{R}_n$ to output the final relative values, ensuring $\sum_q \mathbb{V}_{\text{rec}}^{n,q} = 1$ for the four child nodes. This whole iterative decoding process is written as:

$$\begin{aligned}
\mathbb{M}_{\text{dec}}^{n+1} &= \mathcal{S}_n(\mathcal{U}_n(\mathbb{M}_{\text{dec}}^{n}), \mathbb{M}_{\text{enc}}^{n+1}); \\
\mathbb{V}_{\text{rec}}^{n,q} &= \mathcal{R}_n(\mathbb{M}_{\text{dec}}^{n,q}) \\
p_{\text{leaf}}^{n,q} &= \mathcal{T}_n(\mathbb{M}_{\text{dec}}^{n,q})
\end{aligned} \tag{3.29}$$

Here, $\mathcal{S}_n$ takes both the upsampled feature $\mathcal{U}_n(\mathbb{M}_{\text{dec}}^{n})$ at the $(n+1)$-th level (upsampling increases n by one) and the skip-link feature $\mathbb{M}_{\text{enc}}^{n+1}$ from the same level of the encoder. This skip-link is inspired by the traditional U-Net [136] architecture, and it makes the neural network more robust to spatial size variations through pooling and upsampling. Without skip links, we have to decode an entire $\mathbb{Q}_r$ from only the last layer feature $\mathbb{M}^0$, which is much more difficult and can end up having a shallow tree.

In summary, the decoding process starts from the coarsest 0-th level and gradually builds $\mathbb{Q}_r$ until reaching $n_{\max} = 20$. If all the nodes are leaves when reaching a level, the decoding process terminates early. In practice, the input and output tree can have different numbers of levels. Note that our neural network only predicts the relative value $\mathbb{V}_{\text{rec}}^{n,q}$ ($= 1$ if q is the root node) for every node q on every level n with respect to its parent node value, and actual absolute values in the trees are reconstructed by unrolling the relative values using a series of multiplications. This hierarchical encoder and decoder architecture efficiently extends U-Net style CNNs to quadtrees that are naturally more sparse than image grids.

Loss Function. We train the network to output accurate quadtree distributions as close to the ground-truth quadtrees as possible. The ground-truth trees are generated in the same way as our input trees $\mathbb{Q}_h$ (Section 3.4.5), by tracing and accumulating a large number of samples until converged (more details in Section 3.6). As a result, for each spatial location, we have its ground-truth quadtree $\mathbb{Q}_{\text{gt}}$ with node type label $\gamma_{\text{leaf}}^{n,q}$ and distribution value $\mathbb{V}_{\text{gt}}^{n,q}$ for each node. Here, $\gamma_{\text{leaf}}^{n,q}$ represents the node type in the ground-truth tree, which is deterministic and binary.

Therefore, we can supervise our network output $p_{\text{leaf}}^{n,q}$ and $\mathbb{V}_{\text{rec}}^{n,q}$ with the ground-truth

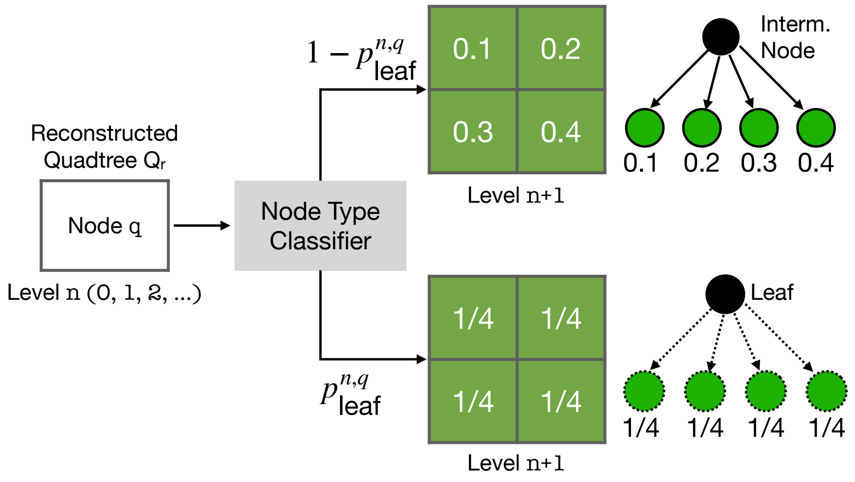


Figure 3.11. Illustration of the loss computation. After the tree Q_r is hierarchically reconstructed by the network, we compute a loss value for every node q at every level n from the top to the bottom. We compute the expected distribution value (Equation 3.31) depending on how probable p is a leaf (i.e., p_{leaf}) predicted by the node classifier $\mathcal{T}_n$. Note that, when p is a leaf, the corresponding distribution values for the next level are just $\frac{1}{4}$.

$\gamma_{\text{leaf}}^{n,q}$ and $\mathbb{V}_{\text{gt}}^{n,q}$ respectively. However, since the ground-truth tree $\mathbb{Q}_{\text{gt}}$ is generated using a lot of samples, its structure can be deep and fine-grained corresponding to a high-resolution distribution; enforcing the network to reconstruct such a deep quadtree structure from sparse input samples is highly challenging and even unrealistic, especially at the beginning of rendering. Therefore, we let the network put more emphasis on regressing accurate distribution values; we seek to allow a different tree structure as long as its final distribution is close to the ground truth. To this end, we focus on the expected distribution value for each node without directly supervising the tree structure.

Given a parent node q and its four potential child nodes q_c, we compute the expectation of the distribution value $\mathbb{V}_{\text{rec}}^{n+1,q_c}$ for each q_c utilizing the node type probability $p_{\text{leaf}}^{n,q}$ of the parent node:

$$\mathbb{E}[\mathbb{V}_{\text{rec}}^{n+1,q_c}] = p_{\text{leaf}}^{n,q} \cdot \frac{1}{4} + (1 - p_{\text{leaf}}^{n,q}) \cdot \mathbb{V}_{\text{rec}}^{n+1,q_c}. \tag{3.30}$$

Note that, our regressed distribution value $\mathbb{V}_{\text{rec}}^{n+1,q_c}$ is a relative value, i.e. a ratio of its actual value to its parent node value. If the parent node q is a leaf node, the distribution is assumed uniform inside the node. Thus, the corresponding relative value for the same region of each q_c is just exactly $\frac{1}{4}$, multiplying $p_{\text{leaf}}^{n,q}$, which is the relative distribution value if q is a leaf and q_c does not exist. We propose to supervise the expected value $\mathbb{E}[\mathbb{V}_{\text{rec}}^{n+1,q_c}]$ with the ground-truth value $\mathbb{V}_{\text{gt}}^{n+1,q_c}$ for all children nodes q_c of q. This loss is given by:

$$\mathbb{L}_{\text{value}}^{n,q} = \sum_{q_c} \|\mathbb{E}[\mathbb{V}_{\text{rec}}^{n+1,q_c}] - \mathbb{V}_{\text{gt}}^{n+1,q_c}\| \tag{3.31}$$

Similar to the above discussion for Equation 3.30, if the ground-truth node q is a leaf ($\gamma_{\text{leaf}}^{n,q} = 1$) and q_c does not exist, we just use $\mathbb{V}_{\text{gt}}^{n+1,q_c} = \frac{1}{4}$. This loss (Equation 3.31 with Equation 3.30) jointly supervises the predicted node type probabilities and the distribution values. However, we find in our experiments that using this loss only can be unstable in the early training time. We therefore provide direct supervision for the tree structure at the beginning of the training, using a binary cross entropy loss $\mathbb{L}_{\text{class}}^{n,q}$ that supervises $p_{\text{leaf}}^{n,q}$ with $\gamma_{\text{leaf}}^{n,q}$. We apply deep supervisions to every generated tree node output on all the levels and sum their losses up. Our full loss function is expressed by

$$\mathbb{L}_{Q_r} = \sum_{n=0}^{n_{Q_r}} \sum_{q \in q_n} (\beta \mathbb{L}_{\text{class}}^{n,q} + \mathbb{L}_{\text{value}}^{n,q}) \tag{3.32}$$

where $\mathbb{L}_{Q_r}$ denotes the loss of the whole reconstructed tree $\mathbb{Q}_r$ summed over every node q on every decoding level n. Here, q_n is the set of nodes on the n-th level, n_{Q_r} is the actual maximum decoding level, and β is a weight factor. During training, we start with $\beta = 1$ in Equation 3.32 to stabilize the early optimization by supervising both the structure (with $\mathbb{L}_{\text{class}}$) and values (with $\mathbb{L}_{\text{value}}$), and then gradually reduce β to zero. Therefore, eventually, we supervise the sampling map implicitly using $\mathbb{L}_{\text{value}}$ without forcing the network to output the same target quadtree structure $\mathbb{Q}_{\text{gt}}$ (an over-strong regularization and often impossible to achieve). Our neural network can generalize well to new scenes since it mainly operates on the local sample input without any strong global scene-level dependency.

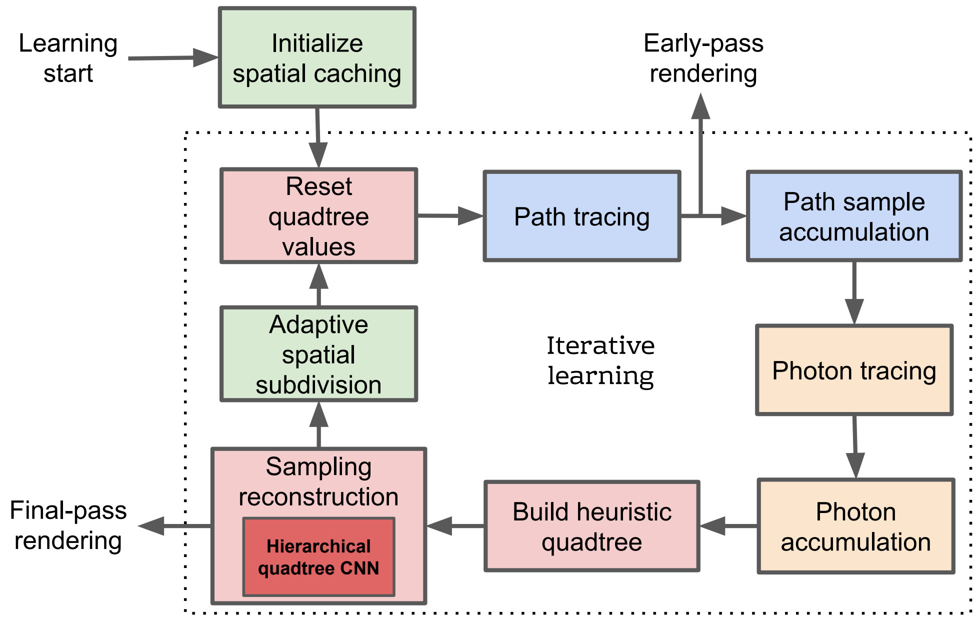


Figure 3.12. Illustration of iterative learning and rendering. We show the pipeline of our path guiding and rendering process (Section 3.4.7). It starts by building a coarse grid $\mathbb{G}$, which is later iteratively refined online. We trace a set of path samples to detect valid spatial voxels in $\mathbb{G}$ for storing sampling distributions, as well as accumulating their input features into per-voxel quadtrees Q_h (Equation 3.25); these path samples also contribute radiance to the rendering result. We then trace photons from the light and deposit them to the corresponding quadtrees Q_h in their arriving spatial voxels. These accumulated quadtrees Q_h are adaptively subdivided (Section 3.4.5) based on the sample information they accumulate in this iteration. We then send Q_h to our neural network and reconstruct accurate quadtrees Q_r, which will be used as sampling distributions to guide the path tracing in the next iteration. Afterward, we refine the voxels of the spatial grid $\mathbb{G}$ as needed. Before moving to the next-iteration path tracing, we reset the values in Q_h to zero, while retaining their tree structure to continue to accumulate samples and possibly obtain further refined quadtrees in the next iteration.

3.4.7 Iterative Learning and Rendering

We use an iterative algorithm to trace and deposit samples, accumulate the initial quadtrees $\mathbb{Q}_h$, reconstruct the accurate quadtrees $\mathbb{Q}_r$ as sampling distributions, and use the learned distributions for rendering the final image. Here, we share a similar design with many state-of-the-art path guiding works [111, 110, 132, 214], as presented in Figure 3.12.

Spatial Caching of Sampling Distributions. We use a hierarchical hash grid $\mathbb{G}$ [214] in the scene to receive the hybrid samples, store the input $\mathbb{Q}_h$ and output $\mathbb{Q}_r$ in individual voxels.

In the first iteration, the traced path samples are collected to determine the bounding box of our spatial grid, which covers the *visible* part of the scene. Next, we start from a discrete 3D volume that uniformly partitions the visible scene space where each voxel is a cube with a side length $\mathbb{R}_\text{B}/r_\text{init}$ where $\mathbb{R}_\text{B}$ is the diagonal length of the initially estimated bounding box; each voxel receives hybrid samples and builds sampling quadtrees, which can be further sub-partitioned to a KD-tree as needed. This leads to a hierarchical spatial grid with per-voxel sampling distributions. Here, each voxel is iteratively subdivided to a KD-tree based on a simple but effective criterion: if the total number of samples N_spt within a spatial voxel is larger than a pre-defined threshold k_spt (i.e., $N_\text{spt} > k_\text{spt}$), then we split the voxel into two sub-voxels through the middle plane along an alternating dimension. This subdivision is applied recursively to sub-voxels until all voxels do not satisfy the subdivision criteria, similar to the strategy proposed by Muller et al. [111]. Therefore, when a new sample arrives, we first search within the hierarchical spatial grid $\mathbb{G}$ to find the voxel that covers the sample, then deposit the sample to its stored quadtree $\mathbb{Q}_h$ (Section 3.4.5). In practice, we also jitter the sample across neighboring spatial voxels (similar to depositing a sample into the angular quadtree in Section 3.4.5), which creates a spatial stochastic box filtering as is done in [110].

Importance Sampling from Quadtree. When guiding paths, the importance sampling of ω_i is done by traversing the $\mathbb{Q}_r$ from the top to the bottom similar to [111, 132]. From the root node of the quadtree, we iteratively sample one from the four child nodes based on their relative value $\mathbb{V}_\text{rec}^{n,q}$, until reaching a leaf node. We then uniformly sample the leaf node. Suppose the leaf has a solid angle bin $\Delta z \cdot \Delta\phi$ in the $(\mathbb{Z}, \Phi)$ space, then the final sampling PDF corresponds to $p_\text{leaf} = 1/(\Delta z \cdot \Delta\phi)$.

Iterative Sample Tracing and Rendering. Similar to most of the previous work [111, 132, 214], we iteratively trace samples (though our samples are uniquely hybrid) to refine our quadtrees over time. Specifically, in the t-th iteration ($t = 0, 1, 2, ..., \mathbb{T}$), we trace 2^t sample-per-pixel (SPP) camera and light rays; each bounce point of the ray yields a path sample or a photon, which is deposited into the quadtree in a corresponding spatial voxel as discussed above.

Each spatial voxel stores an input online accumulated quadtree $\mathbb{Q}_h$ and a neurally reconstructed quadtree $\mathbb{Q}_r$ for path guiding. After each iteration, we reconstruct a new $\mathbb{Q}_r$ from the current $\mathbb{Q}_h$; the values of the input quadtree $\mathbb{Q}_h$ are then cleared to zero, and $\mathbb{Q}_h$ continues to accumulate new samples in the next iteration while inheriting the same tree structure.

After $\mathbb{T}$ iterations, we discontinue learning distributions and initiate a final pass where we use the most recent reconstructed quadtrees $\mathbb{Q}_r$ from the $(\mathbb{T})$-th iteration for guiding the rest of the path samples. Since our approach allows the learning to stop earlier because of high-quality reconstructed distributions, we can save more samples for the final-pass rendering. The final rendered image combines the radiance of all samples from $t \geq 2$ iterations weighted by the inverse of their estimated per-pixel variances [110].

Combining Sampling Strategies. The learned guiding sampling is combined with BSDF sampling via the one-sample MIS (Equation 3.20). In the iterative process ($t < \mathbb{T}$), we use $\alpha = 0.5$ for the one-sample MIS. For the final rendering pass, we follow [214] to compute the blending coefficient α adaptively: $\alpha = \mathbb{E}_{\omega_o,\omega_i}[L^{\text{BSDF}}_{\omega_o,\omega_i}]/(\mathbb{E}_{\omega_o,\omega_i}[L^{\text{BSDF}}_{\omega_o,\omega_i}] + \mathbb{E}_{\omega_o,\omega_i}[L^{\text{guide}}_{\omega_o,\omega_i}])$. Here, $\mathbb{E}_{\omega_o,\omega_i}[\cdot]$ is the expected radiance sent back to the viewing direction using one of the two sampling strategies, which is statistically estimated from the previously traced path samples in past iterations.

3.4.8 Dataset Preparation for Training Quadtrees

We create a large-scale dataset to train our neural network. We collect 50 complex indoor and outdoor scenes either used by researchers in previous papers [175, 111, 132, 6, 214] or designed by artists from online resources [9, 74, 108, 164, 109, 107]. Following [212, 214], we also add additional procedurally generated training scenes, created by combining multiple randomized geometry primitives under area lights and environment maps. We hold out 12 (from the 50) complex scenes as testing scenes to evaluate our method. The remaining scenes are used for training.

We use the same method (described in Section 3.4.5) to create the input (noisy) and

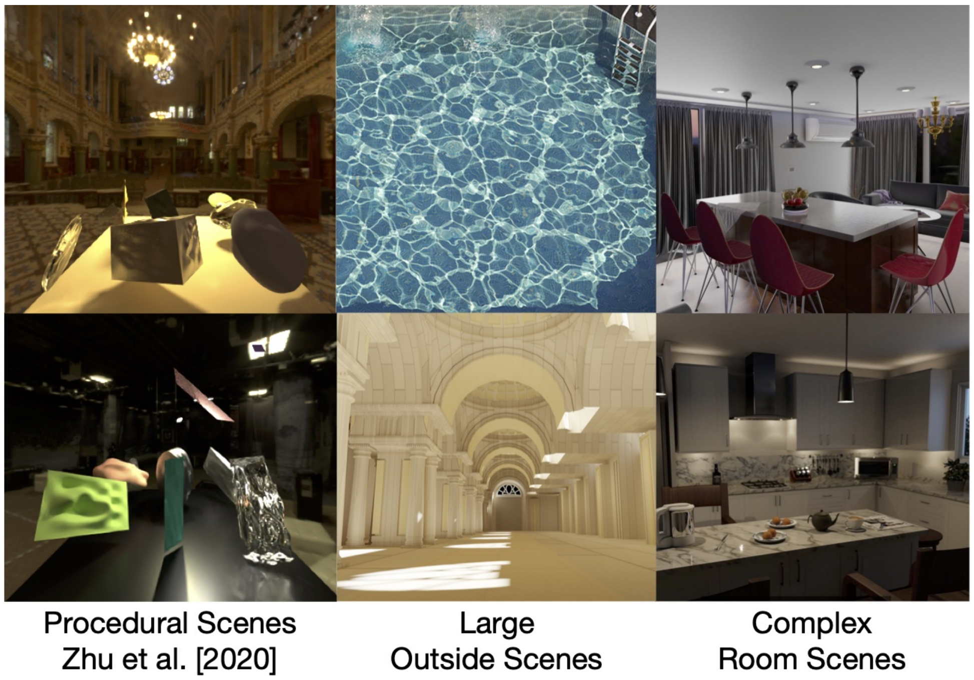


Figure 3.13. Multiple sets of training scenes, including diverse procedural random scenes from the previous work [214] and complex indoor and outdoor scenes designed by researchers and modeling artists.

output (ground-truth) quadtrees from the training dataset. In particular, we iteratively emit 2^t SPP camera and light rays in iteration $t \in [0, \mathbb{T}]$ to create path and photon samples and accumulate them in the spatial grid $\mathbb{G}$ in the scene, similar to the rendering process in Section 3.4.7. We obtain the input quadtrees $\mathbb{Q}_h$ by accumulating hybrid samples in the spatial voxels at every $t_{\text{in}} \in [0, 12]$ iteration. For every input $\mathbb{Q}_h$ at iteration t_{in}, we freeze the spatial cache $\mathbb{G}$ for the following iterations $t > t_{\text{in}}$, continue collecting more samples and repeatedly refine $\mathbb{Q}_h$ to create the $\mathbb{Q}_{\text{gt}}$ when reaching $t = t_{\text{gt}} = 20$. When accumulating hybrid samples, we use the BSDF marginalized variance-aware sampling function (Equation 3.22) for the path samples, unless otherwise stated in ablation studies (Section 3.7). As for photons, we simply use their power (Equation 3.23) as the input values.

We also apply additional data augmentation designs to increase the generalization ability of the neural networks. Since the hierarchical hash grid $\mathbb{G}$ has KD-trees $\mathbb{B}_{\text{spt}}$ containing spatial voxels of different sizes (Section 3.4.7), we augment the training data by selecting 10 different initial resolutions r_{init} equally spaced between $r_{\text{init}}^{\text{min}} = 10$ and $r_{\text{init}}^{\text{max}} = 200$, which can cover diverse voxel sizes. We also further augment the input by randomly rotating the global frames.

3.5 Light Path Guiding Extension

In this work, we leverage photons to generate our sampling distributions. However, it is well-known that tracing photons can sometimes be inefficient, especially when only a small region of an enormous scene is visible to the camera; many traced photons may never reach any valid voxels, leading to expensive photon tracing and undesirably overly-dense spatial structure. Guiding the tracing of photons can address this issue to some extent. We adopt a simple extension of photon guiding by applying the light path guiding technique similar to [175], which improves the rendering quality of scenes that are difficult for the standard photon tracing.

3.6 Neural Path Guiding Implementation

In this section, we discuss some details in neural network training and rendering setup.

Neural Network Training. Our network architecture is designed to be compact for fast inference in rendering. The maximum number of feature channels in our neural network is set to be 128. While this leads to efficient sampling reconstruction, it is still challenging for such a single network to handle diverse inputs with various numbers of input samples or very different sparsity levels. Therefore, we train five separate versions of the same network as is done in [214], where each one only needs to handle the input $\mathbb{Q}_h$ that contains a certain range of sample numbers (i.e., $[0,100)$, $[100,500)$, $[500,1000)$, $[1000,5000)$, $[5000,\infty)$). During both training and testing, we split the set of $\mathbb{Q}_h$ into these smaller groups, and these networks are executed on GPUs in parallel to reconstruct the set of $\mathbb{Q}_r$. We train these networks using the ADAM optimizer [88] with a learning rate of 1.0×10^{-4} until convergence.

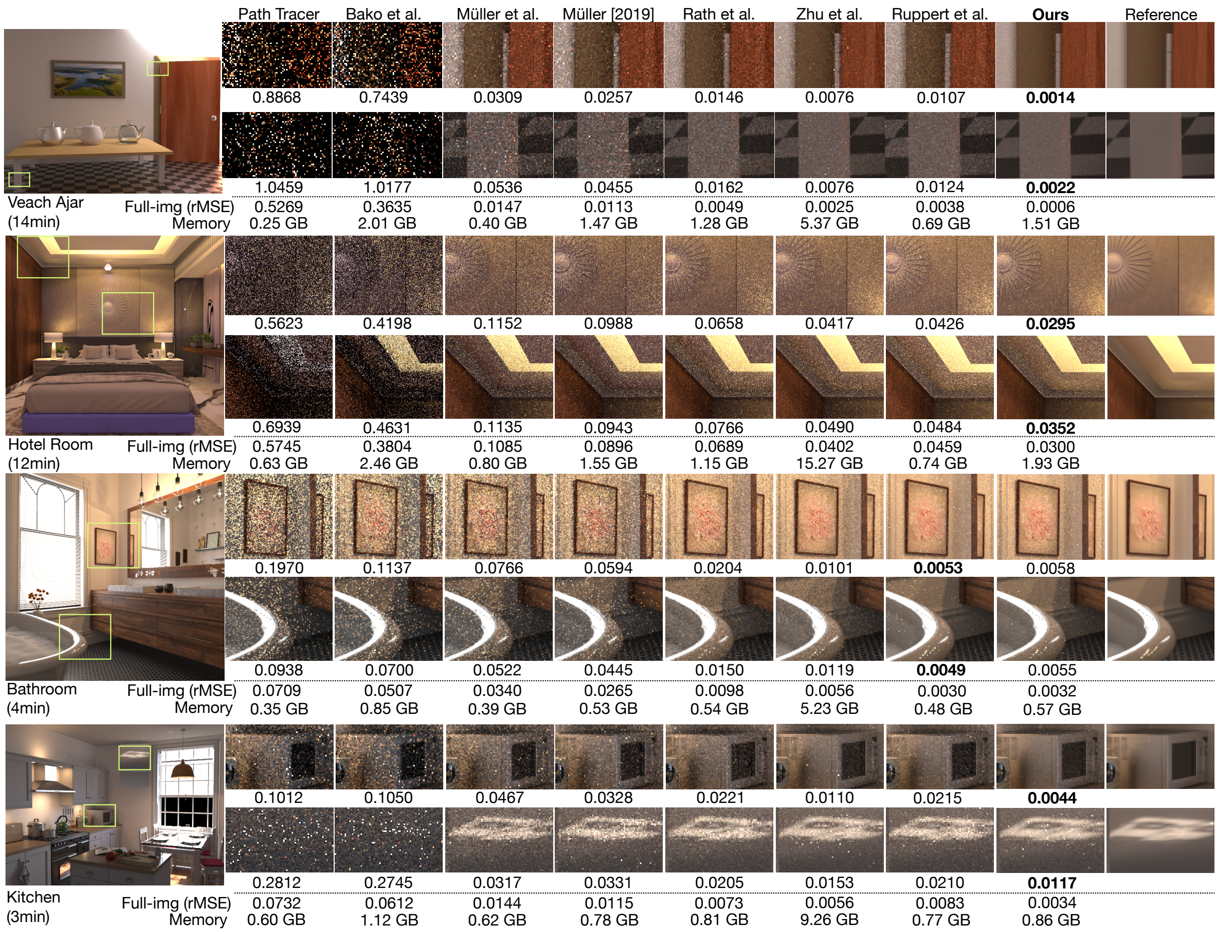
Rendering. When rendering, we stop learning distributions after $5 \sim 10$ iterations depending on the actual light transport complexity of each scene, and guide the remaining path samples in the final-pass rendering. Experiments are rendered on a workstation with an Intel Core i9-7960X CPU and two Nvidia Titan RTX GPUs required to run our neural networks. For some simple testing scenes, one GPU is sufficient. Sample tracing and rendering are performed on the Mitsuba engine [74]. The neural network is integrated into the rendering engine using the TensorFlow C++ API with acceleration libraries and other standard C++/CUDA libraries for efficient data streaming. To utilize the potential parallelization between the CPU and GPUs, the CPU keeps ray tracing and rendering the current-pass result using the previously reconstructed sampling distributions until the GPU finishes computing a new set of $\mathbb{Q}_r$ and updating those distributions. This effectively keeps the CPU and GPU running busy and staying at high utilization. Our quadtree-based neural networks are efficient to evaluate. The GPU processing time is about $6\% \sim 15\%$ (varying across scenes) of the CPU processing time in our experiments. In the future, implementing our proposed neural path guiding framework into a GPU-based rendering engine leveraging hardware ray-tracing (e.g., [122]) can possibly result in higher efficiency in practice.

3.7 Experimentation and Evaluation

We present extensive evaluation in this section. Since our latest attempt (Section 3.4) works better compared to our initial design (Section 3.3), in this section, we denote the first published work [214] as one of baselines and the superior model [215] as the primarily proposed approach.

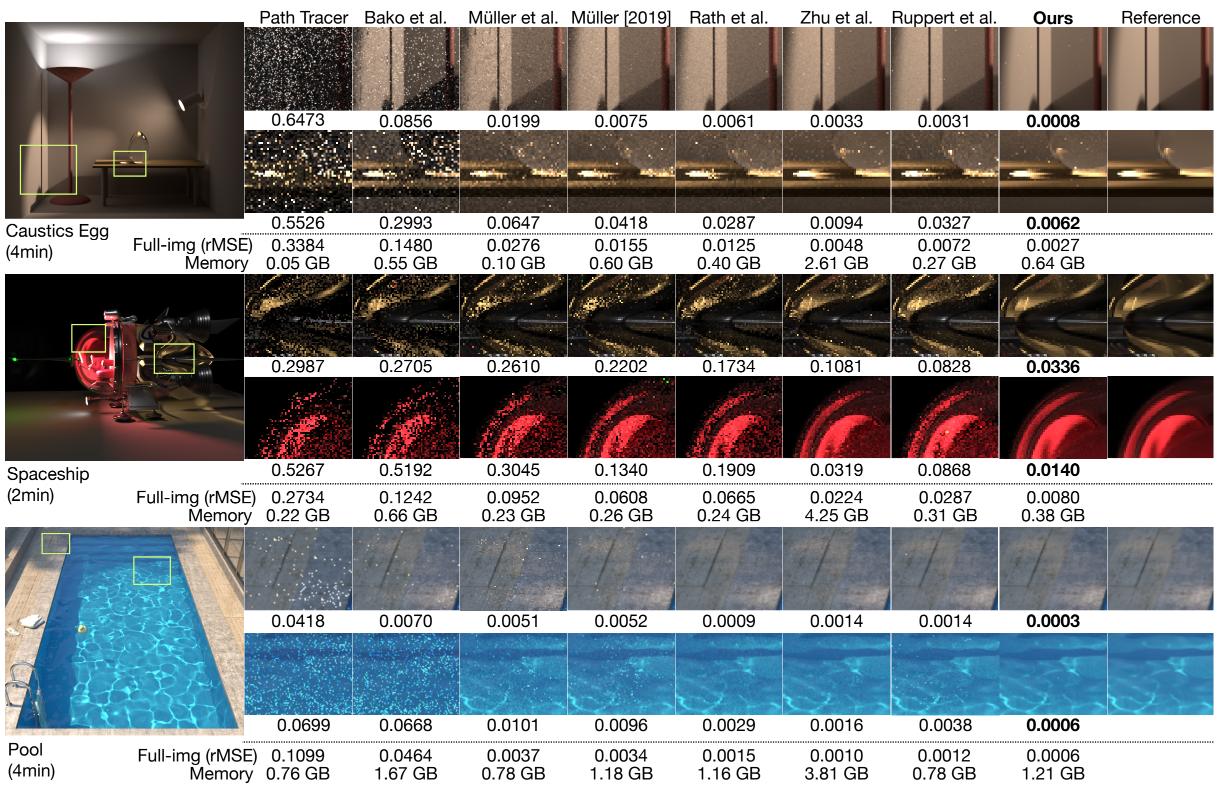
3.7.1 Light Transport Configuration

We evaluate our method on 12 complex testing scenes, each containing complex global illumination and diverse geometric variations. When rendering each scene, we limit the maximum number of bounces to 20; Next Event Estimation (NEE) is turned off (except for Figure 3.9) to clearly show the effectiveness of path guiding for ours and all comparison methods. We compare our methods with several traditional online path guiding methods [111, 110, 132, 138] which do



		Path Tracer	Bako et al.	Müller et al.	Müller [2019]	Rath et al.	Zhu et al.	Ruppert et al.	**Ours**	Reference
Veach Ajar (14min)		0.8868	0.7439	0.0309	0.0257	0.0146	0.0076	0.0107	**0.0014**	
		1.0459	1.0177	0.0536	0.0455	0.0162	0.0076	0.0124	**0.0022**	
	Full-img (rMSE)	0.5269	0.3635	0.0147	0.0113	0.0049	0.0025	0.0038	0.0006	
	Memory	0.25 GB	2.01 GB	0.40 GB	1.47 GB	1.28 GB	5.37 GB	0.69 GB	1.51 GB	
Hotel Room (12min)		0.5623	0.4198	0.1152	0.0988	0.0658	0.0417	0.0426	**0.0295**	
		0.6939	0.4631	0.1135	0.0943	0.0766	0.0490	0.0484	**0.0352**	
	Full-img (rMSE)	0.5745	0.3804	0.1085	0.0896	0.0689	0.0402	0.0459	0.0300	
	Memory	0.63 GB	2.46 GB	0.80 GB	1.55 GB	1.15 GB	15.27 GB	0.74 GB	1.93 GB	
Bathroom (4min)		0.1970	0.1137	0.0766	0.0594	0.0204	0.0101	**0.0053**	0.0058	
		0.0938	0.0700	0.0522	0.0445	0.0150	0.0119	**0.0049**	0.0055	
	Full-img (rMSE)	0.0709	0.0507	0.0340	0.0265	0.0098	0.0056	0.0030	0.0032	
	Memory	0.35 GB	0.85 GB	0.39 GB	0.53 GB	0.54 GB	5.23 GB	0.48 GB	0.57 GB	
Kitchen (3min)		0.1012	0.1050	0.0467	0.0328	0.0221	0.0110	0.0215	**0.0044**	
		0.2812	0.2745	0.0317	0.0331	0.0205	0.0153	0.0210	**0.0117**	
	Full-img (rMSE)	0.0732	0.0612	0.0144	0.0115	0.0073	0.0056	0.0083	0.0034	
	Memory	0.60 GB	1.12 GB	0.62 GB	0.78 GB	0.81 GB	9.26 GB	0.77 GB	0.86 GB	

Figure 3.14. Equal-time comparisons. We compare our method with previous path guiding methods [111, 110, 6, 132, 214, 138] on complex indoor scenes. For each scene, we show visual comparisons on two crops with corresponding rMSE numbers. We also show the rMSE of the full image and the memory usage for all the methods. Our approach often achieves better visual quality and lower rMSE (on both crops and full images). Our method achieves this with memory cost that is comparable to traditional methods [110, 132] and much less than the previous neural technique [214].



		Path Tracer	Bako et al.	Müller et al.	Müller [2019]	Rath et al.	Zhu et al.	Ruppert et al.	**Ours**	Reference
Caustics Egg (4min)		0.6473	0.0856	0.0199	0.0075	0.0061	0.0033	0.0031	**0.0008**	
		0.5526	0.2993	0.0647	0.0418	0.0287	0.0094	0.0327	**0.0062**	
	Full-img (rMSE)	0.3384	0.1480	0.0276	0.0155	0.0125	0.0048	0.0072	0.0027	
	Memory	0.05 GB	0.55 GB	0.10 GB	0.60 GB	0.40 GB	2.61 GB	0.27 GB	0.64 GB	
Spaceship (2min)		0.2987	0.2705	0.2610	0.2202	0.1734	0.1081	0.0828	**0.0336**	
		0.5267	0.5192	0.3045	0.1340	0.1909	0.0319	0.0868	**0.0140**	
	Full-img (rMSE)	0.2734	0.1242	0.0952	0.0608	0.0665	0.0224	0.0287	0.0080	
	Memory	0.22 GB	0.66 GB	0.23 GB	0.26 GB	0.24 GB	4.25 GB	0.31 GB	0.38 GB	
Pool (4min)		0.0418	0.0070	0.0051	0.0052	0.0009	0.0014	0.0014	**0.0003**	
		0.0699	0.0668	0.0101	0.0096	0.0029	0.0016	0.0038	**0.0006**	
	Full-img (rMSE)	0.1099	0.0464	0.0037	0.0034	0.0015	0.0010	0.0012	0.0006	
	Memory	0.76 GB	1.67 GB	0.78 GB	1.18 GB	1.16 GB	3.81 GB	0.78 GB	1.21 GB	

Figure 3.15. Equal-time comparisons. Similar to Figure 3.14, we show more equal-time comparisons between our method and previous path guiding methods [111, 110, 6, 132, 214, 138]. Our method can also achieve better qualitative and quantitative results using moderate memory costs.

not leverage deep learning techniques (CPU-only) but either use hierarchical quadtrees (similar to ours) or mixture models as their sampling distribution representation. We also compare with neural guiding methods, including one [6] that can only guide the first bounce, and a recent photon-driven approach [214] that can guide multiple bounces; these previous neural methods represent sampling distributions as regular images. For quantitative results, we use the standard relative Mean Squared Error (rMSE) widely used in previous work [132, 214]. All the numbers are computed on tone-mapped LDR images. In addition, we also show the memory cost of each method.

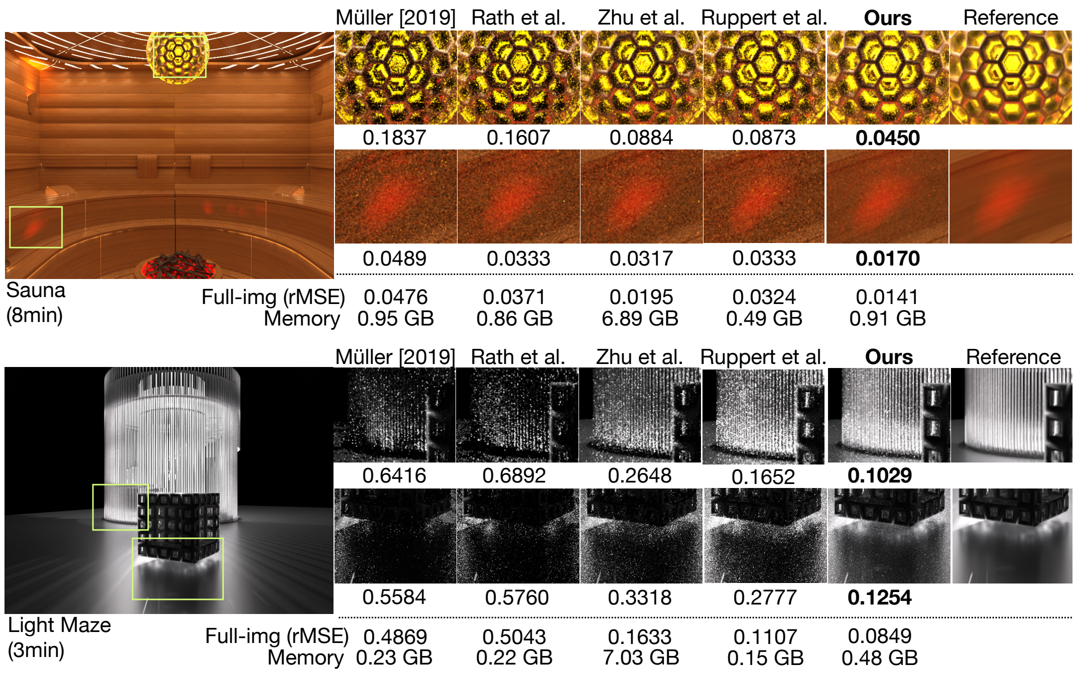


Figure 3.16. Equal-time comparisons with some best performing baseline methods [110, 132, 138] on two complex-visibility scenes. The incident radiance fields of these scenes contain high-frequency details and repeated patterns. We can still achieve better results in such light transport scenarios.

3.7.2 Qualitative and Quantitative Comparisons

Figure 3.14, 3.15 and 3.16 show equal-time comparisons between our method and previous path guiding methods on various complex (indoor, outdoor, and object) scenes. Note that our approach often achieves better qualitative and quantitative results. Our results of zoomed-in rendering crops are smoother, showing less noticeable noise than other results, and are visually closer to the reference. In contrast, the previous first-bounce guiding method [6] cannot handle these challenging cases very well, although it also leverages deep learning techniques; it can only improve the primary bounce sampling thus performs worse than the other guiding methods including the traditional online ones on our testing scenes with strong indirect illumination. The three traditional methods [111, 110, 132] use pure path samples as input and reconstruct hierarchical quadtree distributions online for multi-bounce path guiding. They achieve effective path guiding and improve over the standard path tracing; in particular, Rath et al. [132] shows clear advantages over the other two because of its more efficient variance-aware sampling distribution. Other than the quadtree, Ruppert et al. [138] leverages mixture models (VMMs) to fit path samples by an online adaptive optimization framework, which outperforms many other techniques due to the careful positioning of mixture components and a novel parallax compensation module. However, these methods still leverage a slow online learning process, requiring a large number of path samples and many iterations to achieve accurate distributions for path guiding. The recent photon-driven neural method [214] uses a pre-trained network to relieve this slow online learning, leading to better results. However, this technique [214] (same to [6]) can only reconstruct sampling distributions as regular 2D images (unlike quadtrees) that have a fixed low resolution, hence restricting the accuracy and efficiency of sampling. Instead, our approach directly regresses hierarchical quadtrees from hybrid samples for sampling and can represent more fine-grained distributions under different light transport conditions. As a result, our approach further outperforms [214].

We achieve better rendering quality without a large memory overhead; the sparseness of our representation and the effectiveness of our neural reconstruction lead to high memory efficiency. The recent neural technique [214] requires much larger memory due to the use of grid representation (image). For most scenes, our memory consumption is comparable to the traditional methods [110, 132] without deep learning.

3.7.3 Effect of Hybrid Samples

To further demonstrate the effectiveness of using hybrid samples, comparisons on two extreme light transport settings are shown in Figure 3.9 earlier in this chapter. These two Cornell Box scenes are specifically designed to make only one type of input samples (either paths or photons) useful. Previous methods that use either path samples or photon samples cannot work effectively on both challenging cases. In contrast, our approach uses a hybrid of both path samples and photons with a novel hierarchical neural reconstruction, leading to more robust rendering in both cases. Our neural network learns to correlate the information and convert it

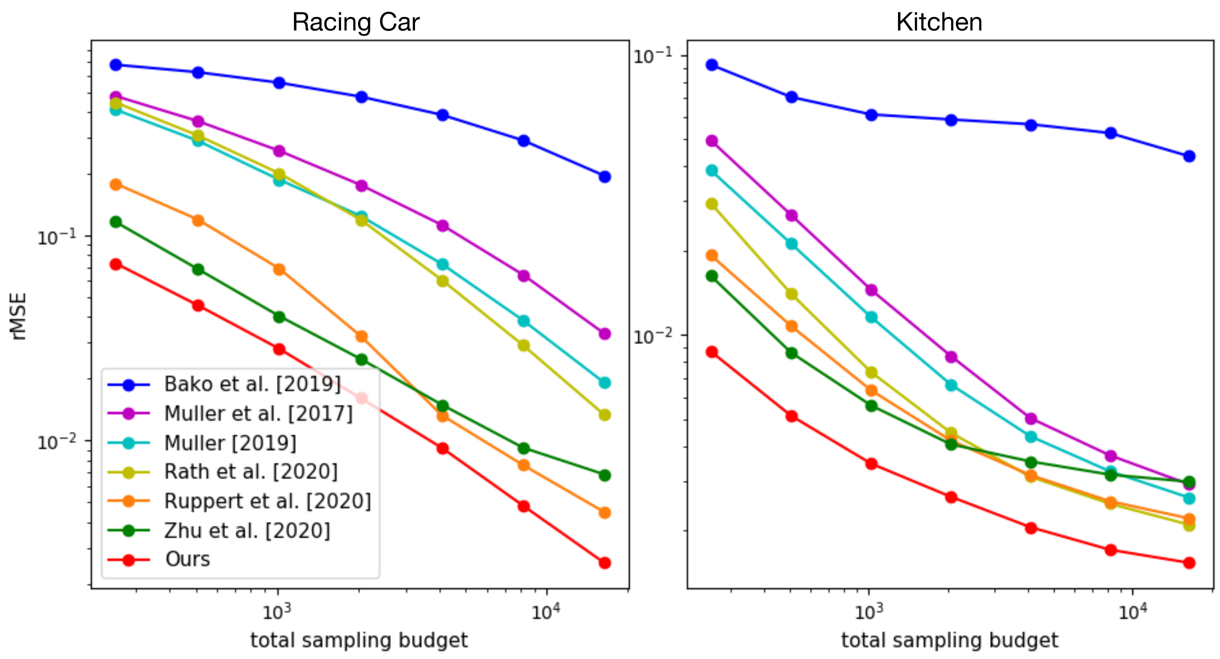


Figure 3.17. Convergence curves of two testing scenes (from 256 SPP to 16,384 SPP). We compare our approach with previous methods using different numbers of samples. The sampling budget represents the total number of rays (including both camera and light) per pixel through the entire guiding and rendering process. Both the X (number of samples) and Y (rMSE) axes are on a logarithmic scale. Our hierarchical neural path guiding performs consistently better with the increasing samples on these two scenes. Because some rays are used for guiding and the quality of the guiding distribution influences the convergence, these curves are not straight lines, as expected for standard path tracing in a log-log plot. Also note that, the recent previous neural method [214] may not be more effective than the traditional methods when using a vast number of samples.

into a single high-quality hierarchical sampling distribution. As demonstrated in other results of complex scenes (Figure 3.7, 3.14 and 3.15), our proposed framework with hybrid input can robustly work well across various challenging light transport cases.

3.7.4 Convergence Rate

We also evaluate how our method performs with an increasing number of samples. In particular, we run our method on two testing scenes (RACING CAR and KITCHEN, shown in Figure 3.7 and 3.14) with different total numbers of traced rays (including both camera and light rays) per pixel and compare the rMSEs with other methods using the same budgets of

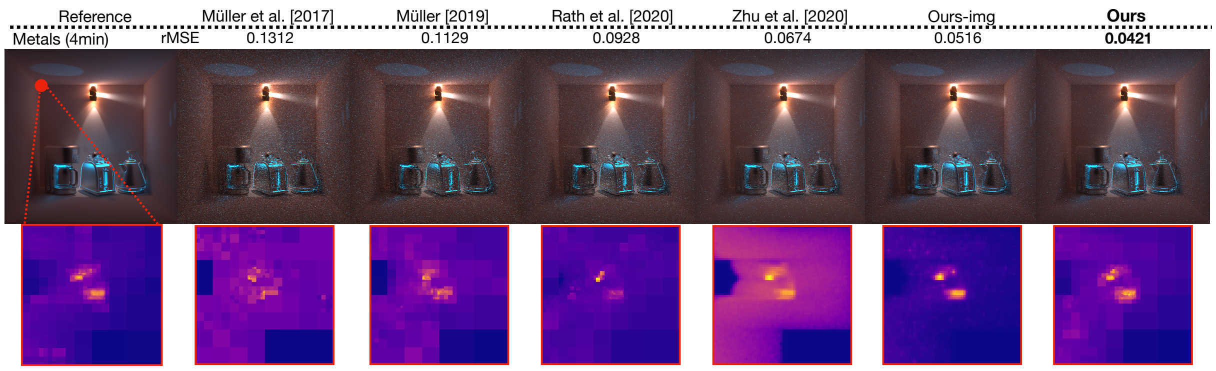


Figure 3.18. Hierarchical reconstruction. We compare with previous methods and show the corresponding sampling distributions of all methods for a scene point (marked by the red point in the reference). Similar to Figure 3.9, the ground-truth sample distribution is with respect to our method and [132] (and also the "Ours-img" variant). We also compare with a non-hierarchical variant (labeled with Ours-img) that takes hybrid samples as input but regresses image grid distributions using the same network architecture as [214]. Our full model leverages hierarchical reconstruction to regress accurate sampling distributions and achieve better results compared to its non-hierarchical counterpart.

sampling rays. The results are shown in Figure 3.17. We can see that our novel neural path guiding approach consistently achieves lower errors with more samples; ours also has smaller errors compared to previous methods. Note that, while the recent neural method [214] can often achieve better results than the other traditional methods with a moderate sampling budget, its gain gets reduced with very large sampling budgets due to the fixed resolution sampling map that intrinsically cannot express the high-frequency lighting perfectly. On the other hand, traditional quadtree-based methods [111, 110, 132] can be more fine-grained with a large number of samples, leading to better results eventually. This example illustrates the benefits of having a hierarchical representation. Our approach successfully applies hierarchical quadtree-based sampling in neural path guiding, leading to efficient rendering.

3.7.5 Hierarchical Representation Merit

We show some examples of the reconstructed sampling distributions in Figure 3.9 and 3.18. Our regressed quadtree distributions are accurate and fine-grained, and are close to the reference. In contrast, [132] is reconstructing the same target distribution as ours, but it leverages traditional online accumulation, which often obtains more noisy quadtrees. Essentially, our neural network is trained to denoise such noisy online-accumulated quadtrees into the smooth and accurate quadtrees. On the other hand, the neural techniques [214] that use uniform grids (images) as the sampling representation can also reconstruct smooth sampling distributions. However, because of the limited image resolution, their sampling is less sharp and detailed compared to our reconstructed quadtrees.

We further investigate the benefits of using the hierarchical network, by training and comparing it with a network that regresses 2D images from hybrid samples without any hierarchical structure. In particular, we use the recent network architecture of [214] and train it using the same hybrid samples as input and the same variance-aware target distribution for path guiding. The corresponding results compared with the results of our full model and other methods are shown in Figure 3.18, with corresponding sampling distributions. This non-hierarchical network with the image representation performs worse than our full model. Meanwhile, our reconstructed hierarchical distribution contains more details and is faster to compute than the uniform image representation, which leads to more efficient path guiding and better rendering quality.

3.7.6 Target Sampling Distribution

Our framework has good flexibility; it can support various target sampling distributions. By default, we use the recent variance-aware function [132] (Equation 3.22) for the better performance. In Figure 3.19, we show results from a variant of our model trained with the traditional target sampling function without variance-awareness (Equation 3.22 as is used in [111, 110]). Note that our approach still works well even without the variance-aware technique, and can still outperform many previous methods. Our full model can achieve better results, taking advantage of the advanced target sampling function.

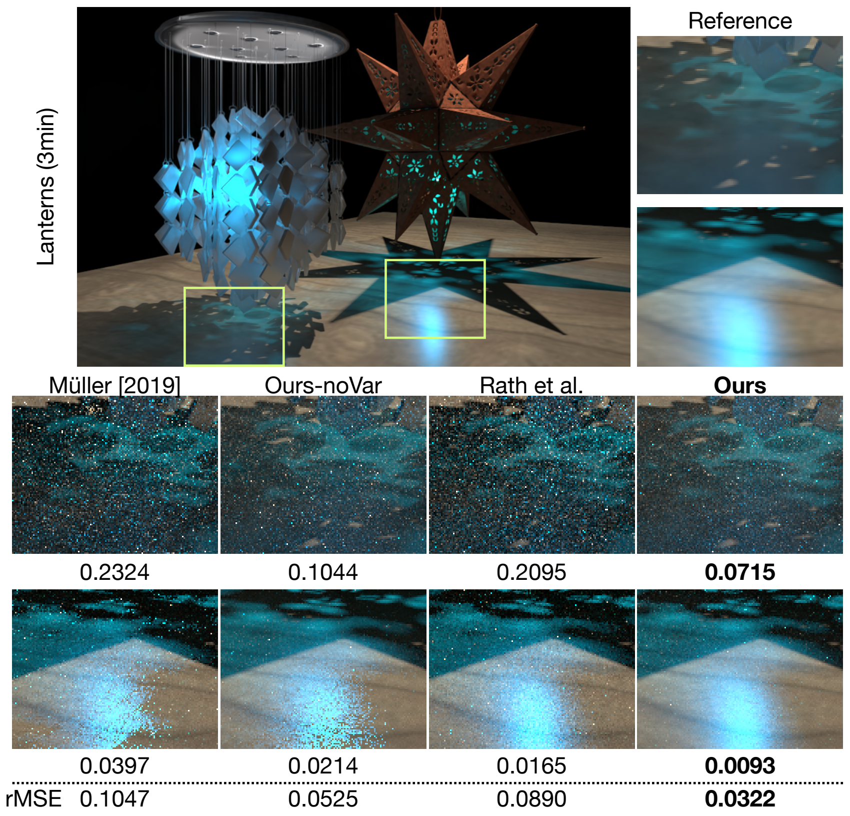


Figure 3.19. Target sampling distribution. Our approach, by default, uses the variance-aware sampling function (Equation 3.22) as the target to train the network. We can also use a simpler target sampling distribution (Equation 3.19) without variance-awareness. We compare this with our default model and also traditional methods using the two different sampling functions. Our full model performs better, which justifies the variance-aware technique and the necessity of using an advanced target distribution for training.

3.7.7 Limitations and Failure Cases

The Image-Based Pure Photon Approach. The proposed approach is mainly designed for offline rendering, as previous path guiding work; it accelerates the convergence of path tracing but still requires a moderate number of path samples. Combining our approach with modern denoising techniques can further reduce the number of samples. Similar to previous work [111, 132] that also use spatial voxels to store local sampling distributions, a structured artifact can appear in the rendered image when photons are not dense enough. Such artifact disappears with more photons; combining our method with parallax-aware techniques [138] through warping or transformation could potentially address it more effectively in the future, but may also expose new challenges in the computational cost of histograms over mixture lobes.

We use standard 2D images as sampling distributions for deep CNN-based reconstruction. However, this consumes more memory than the quadtrees in [111, 132] and parametric models in [175, 138], and we can only adopt low-resolution histogram maps due to limited system memory. The other more compact representations can in fact fit more detailed sampling distributions due to their adaptive nature, although this also requires many more samples. We observe that previous methods start to overtake ours with a very large sampling budget (more than $10^3 \sim 10^4$ rays per pixel). However, our approach is still effective with a moderate sampling budget, which is often how path guiding is expected to be applied, especially when it can be effectively combined with denoising techniques in practice. Extension to more compact directional representations is possible, as shown in a recent concurrent work [215].

Tree-Based Hierarchical and Hybrid Approach. Our approach leverages path and photon samples and treats them equally, tracing the same number of rays for each type of sample in guiding and rendering. However, the two types of samples often do not contribute equally to the final distribution (as in Figure 3.9) and one of them can be less useful, which is a waste of the sampling budget. Addressing this may require future research to support distributing the samples non-equally, adaptive to the actual light transport cases. Besides, we believe guiding the photon emission and tracing properly (e.g., [175]) can be very useful to our framework, which reduces the well-known photon visibility issue and further increases the robustness of our approach.

Our framework utilizes discrete voxels to partition the scene and cache the sampling distributions. Similar to previous methods that use similar caching techniques [111, 214], this spatial structure can have discontinuous sampling distributions across neighboring voxels, leading to some aliasing artifacts that are usually gone after a number of iterations. While our neural framework accelerates the convergence of sampling reconstruction, which alleviates this issue to some extent, some minor artifacts can still appear in early iterations. Exploring an idea similar to the parallax compensation [138] is left for future work.

Currently, we implement our approach in a hybrid CPU and GPU fashion where tracing/shading and sampling reconstruction are executed separately. The extra data copying overhead is still non-negligible even if we carefully manage the data flow and parallelization. In practice, it can be beneficial to put more modules on GPUs directly and make use of the specialized processor cores.

3.8 Conclusion and Future Work

In this chapter, we first present a new deep learning-based photon-driven path guiding approach. Our approach leverages photons to reconstruct sampling distributions, which is sometimes more effective than pure unidirectional (path-driven) methods for challenging scenes that are dominated by indirect lighting; we propose to use a deep neural network to regress high-quality sampling maps from low-quality photon histograms, enabling effective path guiding as a result. To better utilize the benefits of such a neural framework, we introduce an adaptive hierarchical grid to cache the reconstructed sampling maps spatially in the scene, allowing for path guiding at any bounce.

Next, we present a novel path guiding framework that is learning-based, hierarchical, and hybrid. We present a unique neural network that extends traditional CNNs to hierarchical representations, and produces accurate sampling distributions faster than traditional online accumulation methods. Our approach further uses a hybrid of path samples and photons as input, allowing for increased robustness and generality across different complex light transport scenarios. We demonstrate extensive experiments on diverse testing scenes. Our proposed neural path guiding framework can achieve state-of-the-art rendering quality with a reasonably small memory cost compared to other existing approaches.

Our approach also inspires interesting future research. In this work, we focus on making the local directional distribution reconstruction neural and hierarchical. Future work can explore if the spatial caching grid can also be hierarchically reconstructed via a neural network, potentially making local distribution reconstruction aware of the global context. Another interesting direction is to combine our offline neural framework with the online neural techniques [112, 113] that regress a global and continuous sampling function. Meanwhile, combining with the adjoint Russian roulette and splitting technique [174] and extending our framework to product sampling can be the immediate next steps. Our approach leverages quadtree-based neural modeling for local light field approximation; this technique can also inspire other related research areas in computer graphics, such as lighting estimation and light transport acquisition.

Chapter Review. In this chapter, we shifted our focus to the multi-bounce global illumination effect on images. By investigating various Monte-Carlo sampling strategies, we have increased the convergence rate of the classic path tracing method.

Acknowledgements. This chapter, in full, is a reprint of two materials as they appear in ACM Transactions on Graphics 2021 [214, 215]. Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi, Association for Computing Machinery, 2021. The dissertation author was the primary investigator and author of these two papers.

Chapter 4

Controlling Scene Appearances on Images

4.1 Photographing Unconventional Illumination Effects

In previous chapters, we have covered both special and generalized illumination effects. But we are not settled for creating only realistic appearances on images. Instead, people have strived to produce more interesting phenomenons since the emergence of digital photography. Such *novel effects* are typically targeted to functionalities that are closely related to specific visual applications.

The representative examples, which will be presented in this chapter, are illustrated in Figure 4.1 and published in [210, 213]. The principle of our system is to *manipulate appearances* for image authentication and prevention of illegal captures. With this in mind, we innovated a few unconventional features, which include the initiation and restoration of banding effects as well as watermark embedding. The final image look is commanded by computerized light sources that are programmable. Following the tradition of computational photography, our design is operated during image formation without requiring access to any post-capture processing.

4.2 System Overview

The ubiquity of mobile camera devices has been triggering an outcry of privacy concerns, whereas privacy protection still relies on the cooperation of the photographer or camera hardware, which can hardly be guaranteed in practice. In this work, we introduce LiShield, which auto-

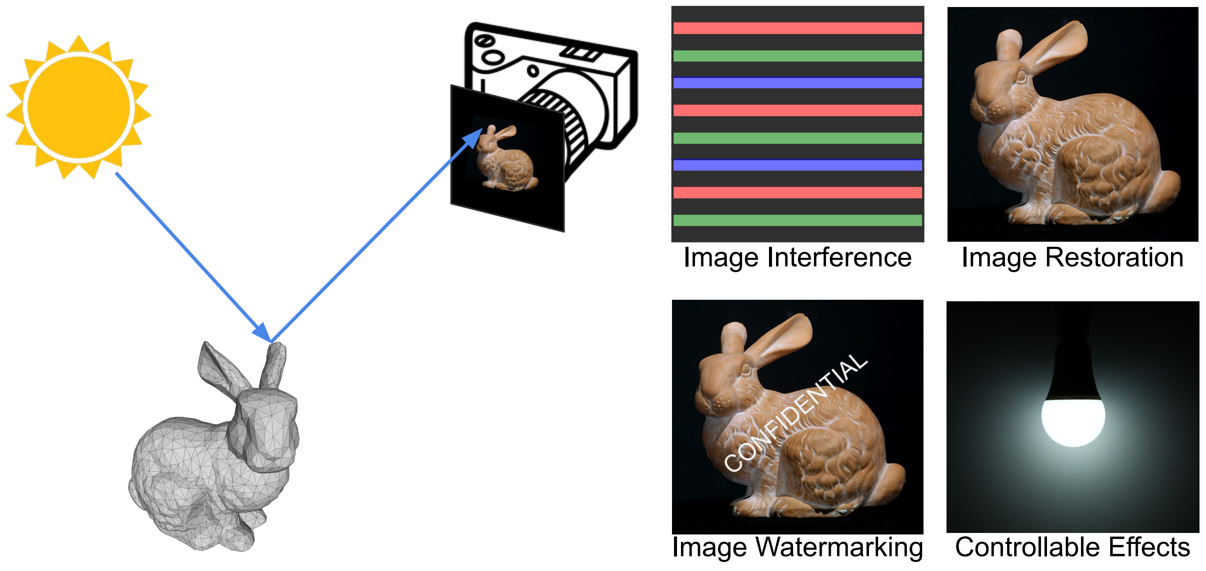


Figure 4.1. Summary of novel effects on images using our computational photography framework. These are extremely difficult to achieve by conventional imaging during the image construction. The objective is to compute those phenomenons well before image processing and editing, which is desirable especially when dealing with ubiquitous and wild captures.

matically protects a physical scene against photographing, by illuminating it with smart LEDs flickering in specialized waveforms. We use a model-driven approach to optimize the waveform, so as to ensure protection against the (uncontrollable) cameras and potential image-processing-based attacks. We have also designed mechanisms to unblock authorized cameras and enable graceful degradation under strong ambient light interference. Our prototype implementation and experiments show that LiShield can effectively destroy unauthorized capturing while maintaining robustness against potential attacks.

4.3 Motivation and Subject of Study

Cameras are now pervasive on consumer mobile devices, such as smartphones, tablets, drones, smart glasses, first-person recorders [115], *etc.* The ubiquity of these cameras, paired with pervasive wireless access, is creating a new wave of visual sensing applications, *e.g.*, autonomous photographer [116], quantified-self (life-logging) [38, 188], photo-sharing social networks, physical-analytics in retail stores [131], and augmented reality applications that navigate users across unknown environment [117, 189]. Zooming in the photo-sharing application alone, statistics report that 350 million photos/videos are uploaded to Facebook every day, the majority of which are from mobile users [151]. Many of these applications automatically upload batches of images/videos online, with a simple one-time permission from the user. While these technologies bring significant convenience to individuals, they also trigger an outcry of privacy concerns.

Privacy is ultimately a subjective matter and often varies with context. Yet many of the privacy-sensitive scenes occur in the indoor environment, and are bound to specific locations. For example, recent user studies [26] showed that people's acceptability of being recorded by augmented reality glasses has a strong correlation with the location. User studies of life-logging cameras [67] also indicate that 70.2% of the cases when the user disables capturing is associated with specific locations. In numerous real-world scenarios, cameras are forbidden, *e.g.*, concerts, theaters, museums, trade shows, hospitals [57], dressing rooms and exam rooms [119], manufacturing plants [8], *etc.* However, visual privacy protection in such passive physical spaces still heavily relies on rudimentary approaches like warning signs and human monitors, and there is no way to enforce the requirements automatically. In personal visual sensing applications like life-logging, even if a user were to disable the camera in private space (*e.g.*, bedroom and washroom), malware could perform remote reconnaissance and targeted visual theft by hijacking the victim's camera [159, 194].

In this work, we propose LiShield, a system that deters photographing of sensitive indoor physical space, and automatically enforces location-bound visual privacy protection. LiShield protects the physical scenes against undesired recording without requiring user intervention, and without disrupting the human visual perception. Our key idea is to illuminate the environment using smart LEDs, which are intensity-modulated following specialized waveforms. We design the waveform in such a way that its modulation pattern is imperceptible by human eyes, but can interfere with the image sensors on mobile camera devices.

More specifically, our basic waveform follows an ON-OFF modulation, which causes the reflection intensity of the scene to "flicker" at high frequency. Digital cameras commonly adopt rolling-shutter image sensors, which sample the scene row by row during capturing. Consequently, LiShield will impose a striping effect on the captured image, as long as its flickering frequency exceeds the camera frame rate. To protect against a wide range of camera settings, we build a numerical model to explore the relation between the image quality degradation and the (uncontrollable) camera configurations (*e.g.*, exposure time). Accordingly, we derive common guidelines to maximize the effectiveness through waveform parameter configurations (*e.g.*, frequency, peak intensity, duty cycle). To further enhance the protection, we take two measures: *(i.)* scramble the color patterns, taking advantage of the array of multi-channel RGB chips commonly available on commercial smart LEDs; *(ii.)* randomize the waveform frequency to counteract exposure time manipulation that may circumvent the striping effect, while ensuring no low-frequency components are generated that affect human perception.

In addition, LiShield can tailor the waveform for two particular use cases: *(i.)* allowing an authorized camera, which shares secret configuration information with the LED to recover the image or video frames it captures. *(ii.)* when strong ambient light interferes with the smart LED, LiShield cannot ensure full protection, but it can still emit structured light which embeds invisible "barcode" into the physical environment. The embedded information can convey a "no distribution" message, allowing online servers (*e.g.*, from Facebook and Instagram) to block and prevent the image from being distributed.

We have implemented LiShield based on a customized smart LED, which allows re-configuration of intensity modulation waveforms on each color channel. Our experiments on real-world scenes demonstrate that LiShield can corrupt the camera capturing to an illegible level, in terms of the image brightness, structure, and color. The impact is resilient against possible post-processing attacks, such as multi-frame combining and denoising. On the other hand, it enables authorized cameras to recover the image perfectly, as if no modulation is present. Even under strong sunlight/flashlight interferences, LiShield can still sneak barcode into the physical scenes which can be decoded with around 95% accuracy.

Preventing all privacy leaks, particularly those by determined attackers with professional global-shutter cameras, is likely impossible. Instead, LiShield aims for preventing ad-hoc capturing from benign camera-phone holders, by simply installing customized smart LEDs to fully cover the target environment. Our main contributions can be summarized as follows:

(i.) Proposing a new concept of automating privacy protection against cameras by modulating an LED's waveforms, and deriving general guidelines for optimizing the waveforms against possible camera settings and image recovery.

(ii.) Designing mechanisms to authorize desired capturing, and to embed protection information into the scene under strong ambient light interference.

(iii.) Verifying the system through a full-fledged testbed implementation and experiments in real environments.

4.4 Related Work

Anti-Piracy and Capturing-Resistant Technologies. Camera recording of copyright screen-displayed videos (*e.g.*, in a movie theater) accounts for 90% of pirated online content [205]. Since screen refresh rate is much higher than video frame rate, Kaleido [205] scrambles multiple frames within the frame periods to deter recording, while preserving viewing experience by taking advantage of human eyes' flicker fusion effects. Many patented technologies addressed the same issue [34, 126, 140, 187, 35, 146, 200, 86, 147, 14, 13, 150, 148, 50, 158]. In contrast, the problem of automatic protection of private and passive physical space received little attention. Certain countries [48, 33] dictate that smartphone cameras must make shutter sound to disclose the photo-capturing actions, yet this does not enforce the compliance, cannot block the photo distribution, and cannot automatically protect against video recording.

Certain optical signaling systems can remotely ban photography in concerts, theaters, and other capturing-sensitive sites. Courteous Glass [80] and a recent Apple patent [163] augment wearable devices with near-infrared LEDs, which are invisible to humans but can be captured by a camera, to convey the hidden privacy appeal of the wearers. These LEDs cannot enforce protection (*e.g.*, through image corruption as in LiShield), and convey information only when they fall in the camera's field of view. BlindSpot [166] adopts a computer vision approach to locate retro-reflective camera lenses, and pulses a strong light beam towards the camera to cause overexposure. Despite its sophistication, the approach fails when multiple cameras coexist with arbitrary orientations.

Invisible Screen-Camera Communications. Recent research also explored novel ways of screen-to-camera visible light communication by hiding information behind the display. VRCodes [190] carries information through high-frequency changes of the selected color, which can be decoded by rolling-shutter cameras. Hilight [98] conveys information by modulating the pixel translucency change in subtle ways. ARTcode [198] embeds data into images by modifying the pixels' colors, which is imperceptible due to human eyes' limited pixel resolution. This line of research is also related to classical watermarking, which hides copyright and authentication information in images/videos through spatial/frequency domain re-encoding [1, 124]. These mechanisms are applicable when the users have full control over the image/video source, but cannot prevent malicious capturing/distribution of physical scenes. On the other hand, conventional luminaries bear natural flickering effects that have been leveraged for localization purposes [209, 204, 203], but the frequencies are too high to cause visible corruption on the camera images.

Privacy Protection for Images/Videos. Conventional visual privacy-protection systems have been relying on post-capture processing. Early efforts employed techniques like region-of-interest masking, blurring, mosaicking, *etc.* [118], or re-encoding using encrypted scrambling seeds [28]. There also exists a vast body of work for hiding copyright marks and other information in digital images/videos [114, 179, 68, 69, 180, 95, 190, 43, 130, 202]. LiShield's barcode protection is inspired by these schemes, but it aims to protect physical scenes prior to capturing.

One common challenge in visual privacy protection is to identify the privacy preference. Location-bound privacy expression can be achieved in everyday life using special signs. Privacy.Tag [10] allows people to express their privacy preference by wearing QR codes. I-Pic [2] allows people to broadcast their privacy preferences using Bluetooth. COIN [206] matches a user's face to a prescribed privacy preference, and can automatically detect and mask people who do not want to be captured. P3 [130] protects photo-sharing privacy by encoding an image into a private, encrypted part and a public, standards-compatible part. PrivacyEye [133] allows a user to manually mark regions on an image that permit access from mobile apps. PlaceAvoider [160] allows first-person camera users to capture and blacklist sensitive spaces *a priori*, and use image matching to block subsequent pictures containing such spaces. These systems only work when the user has complete control over the camera.

4.5 Image Manipulation Model and Appearance Goal

LiShield's end goal is to prevent camera recording in protected indoor physical areas without affecting normal human perception. The scene can be static or dynamic. In either case, we assume one or multiple LiShield-enabled smart LEDs can cover the whole area, while providing illumination similar to normal office lighting without human-perceptible flickering. Whereas conventional lighting and sunlight may co-exist with LiShield's smart LEDs (as to be verified in our experiments), covering the entire target scene with LiShield will ensure the strongest protection.

Now consider an unauthorized user (attacker) who wants to take pictures or videos within the protected space, with cameras and flashes embedded in smartphones, but no professional equipment such as global shutter cameras, filters or tripods. The attacker has full control over the camera parameters (*e.g.*, exposure time, capturing time, white-balancing), and can run any post-processing on the captured images. Nonetheless, with LiShield's protection, the image frames are corrupted, so that a major fraction of each frame is either blank or overexposed while colors are distorted (Section 4.7), which deters image viewing/sharing. In addition, LiShield should maintain its protection while allowing authorized users to capture the same scene simultaneously without distortion (Section 4.8). In case strong ambient interference may degrade LiShield's protection, LiShield embeds barcodes in images/videos captured by the attacker to convey privacy policies and ensures they are detectable even after common post-processing (Section 4.9).

4.6 Primer on Restructuring Image Segments

Cameras and human eyes perceive scenes in fundamentally different ways. Human eyes process continuous vision by accumulating light signals, while cameras slice and sample the scene at discrete intervals. Consequently, human eyes are not sensitive to high frequency flickers beyond around 80 Hz either in brightness or chromaticity [4, 79, 167, 205], while cameras can easily pick up flicker above a few kHz [92, 203]. Equally importantly, human eyes perceive brightness in a non-linear fashion [155], which gives them a broad dynamic range, while cameras easily suffer from overexposure and underexposure when signals with disparate intensities mix in the same scene [134].

Unlike professional or industrial cameras which may have global shutters that mimic human eyes to some degree, nearly all consumer digital cameras, pinhole cameras, and smartphones use the rolling shutter sampling mechanism [100, 129], which is the main contributor to their high-frequency sensitivity. When capturing an image frame, a rolling shutter camera exposes each row sequentially.

LiShield harnesses the disparity between cameras and eyes to disrupt the camera imaging without affecting human vision. It modulates a smart LED to generate high-frequency flickering patterns. The reflection intensity (or brightness) of the target scene also flickers following the same pattern as the LED's illumination, albeit at reduced intensity due to reflection loss. LiShield uses the On-Off Keying (OOK) as the basic modulation waveform (Figure 4.2), which does not require complicated analog front-ends and is widely supported by smart LEDs [42, 41]. Due to rolling-shutter sampling, the rows of pixels that are fully exposed in the ON period will be bright, and rows in the OFF period become dark, thus causing striped patterns on the captured image (Figure 4.2(a)(b)). Partially exposed rows experience moderate brightness. Meanwhile,

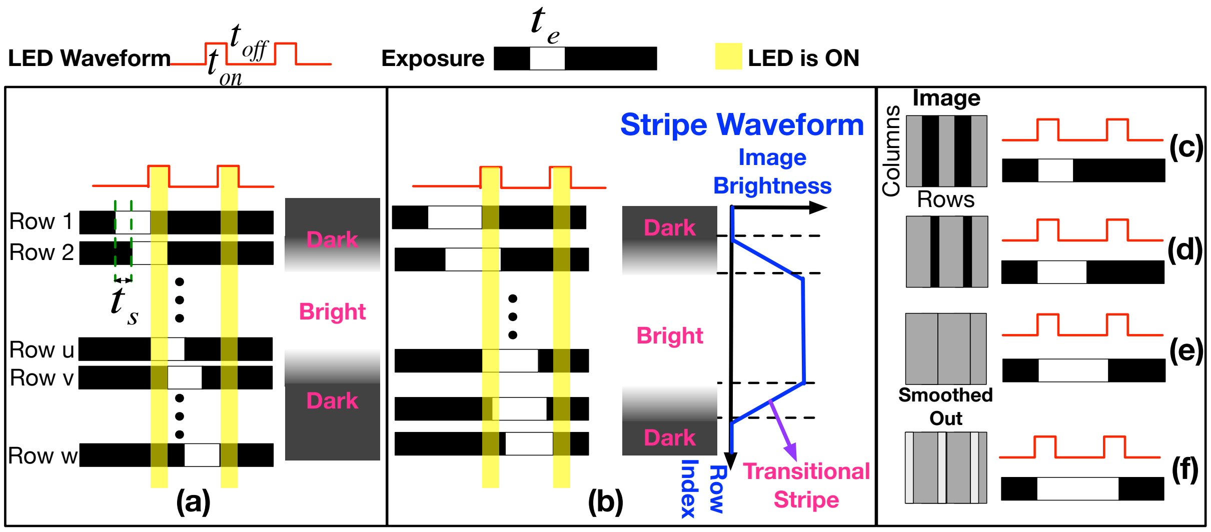


Figure 4.2. (a)-(b) Bright, dark and transitional stripes and their width changing with exposure time; (c)-(f) Stripe pattern of image changes under different exposure times.

human eyes can only perceive the smooth averaged intensity, as long as the OOK frequency goes beyond 80 Hz [4, 79, 167, 205].

In addition, commercial LED fixtures often comprise multiple LED bulbs/chips and sometimes separate RGB channels to allow color adjustments [125]. LiShield can turn different numbers of LED bulb/chip on to generate different intensities, and control the RGB channels of the LEDs to vary the color. Therefore, LiShield's flickering waveform is staircase-shaped on-off patterns, running independently in 3 color channels. In what follows, we will show how such flickering corrupts the spatial patterns captured by a camera.

4.7 Digital Imaging with Coded Illuminations

4.7.1 Maximizing Image Appearance Interference

LiShield aims to minimize the image capturing quality by optimizing the LED waveform, characterized by modulation frequency, intensity, and duty cycle. To explore the optimization space and to provide guidelines for designing the basic waveform, we derive a model to predict the image quality as a function of the LiShield's waveform and attacker's camera parameters. For simplicity, we start with monochrome LED (equivalent to one with a single color channel) that illuminates the space homogeneously. We denote P as the reference image taken under a non-flickering LED, and Q as the one taken under LiShield's LED with the same average brightness. We assume each image has m rows and n columns, and the light energy received by each pixel is denoted by $P(i,j)$ and $Q(i,j)$, respectively. Our model focuses on two widely adopted image quality metrics: *PSNR*, which quantifies the disruption on individual pixel intensity levels; and *SSIM* [185], which measures the structural distortion to the image (*i.e.*, deformation effects such as stretching, banding, and twisting). In general, the minimum PSNR and SSIM corresponding to acceptable viewing quality are in the range of 25$\sim$30 and 0.8$\sim$0.9, respectively [7, 16, 49, 3].

Decomposing the Image. To compute the image quality, we need to model the intensity and width of each stripe caused by LiShield. As illustrated in Figure 4.2, we use $t_{\text{on}}, t_{\text{off}}, I_p$ to denote the on/off duration and peak intensity of the flickering light source, and t_e, t_s are the exposure time (controllable by software) and sampling interval (fixed in hardware) of the rolling shutter camera. For convenience, denote the period of the light source as $t_l = t_{\text{on}} + t_{\text{off}}$, and duty cycle as $D_c = t_{\text{on}}/t_l$. For pixel j in row i which starts exposure at time t_i, its light accumulation would be:

$$Q(i,j) = \alpha_{i,j} \int_{t_i}^{t_i+t_e} \pi_l(\tau) d\tau \tag{4.1}$$

where $\alpha_{i,j}$ is the aggregated path-loss for pixel (i,j), including attenuation and reflection on the photographed object, and $\pi_l(\tau)$ represents the illumination waveform of the LED:

$$\pi_l(\tau) = \begin{cases} I_p, & 0 < \tau \bmod t_l \leqslant t_{\text{on}} \\ 0, & t_{\text{on}} < \tau \bmod t_l \leqslant t_l \end{cases} \tag{4.2}$$

When the camera's exposure time is equal to or shorter than the LED's OFF period ($t_e \leqslant t_{\text{off}}$), the image will contain rows that are completely dark (Figure 4.2(c)). On the other hand, when $t_e > t_l$, one row-exposure period of the camera will overlap multiple ON periods of the LED, accumulating higher intensity (Figure 4.2(f)). The special case happens when $t_e = t_l$ where the integration of LED waveform and exposure has a fixed value, which eventually smooths out dark stripes (Figure 4.2(e)). Without loss of generality, assume the exposure starts right at the beginning of the ON period. Let $N = \lfloor t_e/t_l \rfloor$ which is the number of whole flicker cycles covered by exposure time, and $t_{\text{rem}} = (t_e \bmod t_l)$ which is the remaining duration after multiple whole cycles, the light accumulation of the brightest rows Q_B is:

$$Q_B(i,j) = \begin{cases} \alpha_{i,j} I_p (N t_{\text{on}} + t_{\text{rem}}), & 0 < t_{\text{rem}} \leqslant t_{\text{on}} \\ \alpha_{i,j} I_p (N+1) t_{\text{on}}, & t_{\text{on}} < t_{\text{rem}} \leqslant t_l \end{cases} \tag{4.3}$$

Since the brightest rows appear when the exposure captures most ON periods possible (*e.g.*, row 2 to row u in Figure 4.2 (a)), and rolling shutter effect converts temporal variation into pixels with sampling interval t_s, the width of Q_B is:

$$W_B = |t_{\text{rem}} - t_{\text{on}}| / t_s \tag{4.4}$$

Likewise, when the exposure captures least ON periods possible (*e.g.*, from row v to row w in Figure 4.2 (a)), we get the darkest rows with light accumulation Q_D:

$$Q_D(i,j) = \begin{cases} \alpha_{i,j} I_p N t_{\text{on}}, & 0 < t_{\text{rem}} \leqslant t_{\text{off}} \\ \alpha_{i,j} I_p (N t_{\text{on}} + t_{\text{rem}} - t_{\text{off}}), & t_{\text{off}} < t_{\text{rem}} \leqslant t_l \end{cases} \tag{4.5}$$

and the width of Q_D is:

$$W_D = |t_{\text{rem}} - t_{\text{off}}| / t_s \tag{4.6}$$

We refer to a collection of consecutive brightest rows as “bright stripe” and consecutive dark rows as “dark stripe”, as shown in Figure 4.2(b). In addition, there exist intermediate rows containing linear intensity transition between dark and bright, referred to as “transitional stripe”.

Meanwhile, if the LED were not flickering and provided the same average brightness, the pixel intensity would be:

$$P(i,j) = \alpha_{i,j} I_p \cdot D_c \cdot t_e \tag{4.7}$$

Since $D_c \cdot t_e$ remains constant within each frame, the image captured under LiShield is equivalent to the original image multiplied by a piecewise function (Equations 4.3, 4.5).

Other standard camera parameters (*i.e.*, ISO, white balance, and resolution) do not affect the structure of the stripe pattern, since they are unrelated to rolling shutters and only affect the average pixel intensity. By default, we assume the attacker sets the ISO to its minimum (usually 100) to suppress noise maximally.

Optimizing the LED Waveform. Since the stripe pattern follows a piecewise function, a closed-form expression of PSNR and SSIM becomes infeasible. We thus use numerical simulation to evaluate the impact of LiShield, based on the above model. We generate the piecewise function with $Q_B(i,j)$, W_B, $Q_D(i,j)$, W_D and multiply it on a reference image to obtain the disrupted image Q just like the process inside real cameras. We use the well-known Lena image as a reference, and stitch the original 512×512 version into a 3264×2448 (8-mega-pixel) image, assuming $t_s = 1/75000$ s, which matches the capability of a Nexus 5 camera. The quality metrics are calculated between the reference image P and LiShield-corrupted image Q, which are set to the same average intensity by scaling pixel values in Q. Note that if P and Q are both overexposed into the same white image, PSNR = ∞ and SSIM = 1 can no longer reflect image quality. Thus, we make P's pixel intensity range infinite, which allows quantifying quality loss caused by overexposure.

By default, we use OOK waveform with frequency $f = 100$ Hz, peak intensity $I_p = 10$ kLx and duty cycle $D_c = 0.5$. We vary one parameter while keeping others to the defaults. Note that the typical light intensity is ~ 700 Lx in office environments (considering energy efficiency), $\sim 5{,}000$ Lx for overcast sky and $\sim 100{,}000$ Lx for sunny days [56]. Our numerical results (Figure 4.3 show a few general trends, which lead to the following design choices for LiShield.

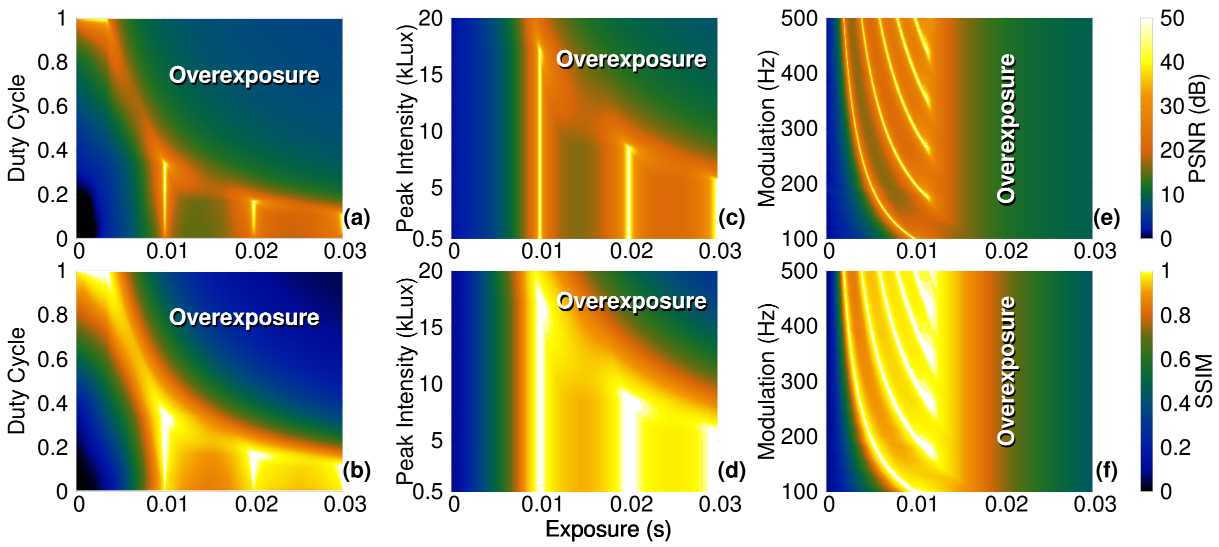


Figure 4.3. PSNR and SSIM with respect to exposure time, LED intensity, duty cycle, and modulation frequency.

(i) A single frequency cannot ensure robust protection. Figure 4.3(e) and (f) show that for a given waveform frequency f, there exist several exposure time settings that lead to high-quality images. This is because when $t_e \approx Nt_l$, the stripes become smoothed out (Figure 4.2(e)). Although the waveform parameters are unknown to the attacker, a determined attacker may launch a brute-force search for the t_e that satisfies this condition, thus circumventing the protection. To counteract such attackers, LiShield includes a countermeasure called frequency randomization, which we discuss in Section 4.7.2.

(ii) LiShield must prevent attackers from using long exposures. The image quality increases with exposure time t_e, until overexposure happens (Figure 4.3(a) and (b)), because longer exposure leads to more waveform cycles being included as a constant base in the brightness of the stripes (larger N in Equation 4.3, 4.5), making the contrast of stripes Q_B/Q_D lower and weakening the quality degradation. Since overexposure limits the maximum exposure time, LiShield should leverage overexposure to limit the attacker's exposure time.

(iii) LiShield should keep a high peak intensity to expand the overexposure zone. We observe that when t_e falls below a threshold ($\approx$1/100 s in Figure 4.3(c) and (d)), the image is always corrupted due to the dominance of dark stripes (Figure 4.2 (c)). On the other hand, when t_e goes beyond a threshold, the image always suffers from overexposure. A larger I_p leads to a smaller overexposure threshold for t_e, which limits the attacker's ability to tune t_e to improve image quality. When $I_p \geq 10$ kLx (Figure 4.3(c)), there almost exists only a single t_e setting ($t_e \approx 1/100$ s) that can avoid overexposure and dark stripes simultaneously. But even this setting fails under LiShield's frequency randomization mechanism (Section 4.7.2). With power efficiency and eye health in mind (Section 4.12), LiShield sets I_p to 20 kLx by default.

(iv) Duty cycle should be kept at a moderate level. Without overexposure, a lower D_c yields lower PSNR and SSIM (Figure 4.3(a) and (b)), as it widens the dark stripes (Equation 4.6). On the other hand, a larger D_c means more light accumulation, resulting in overexposure across a wider range of t_e settings. Since higher I_p has the same impact given a fixed D_c, we design the LED waveform to have maximum peak intensity with moderate duty cycle, empirically set to $D_c = 0.5$.

The above conclusions hold for all scenes since the trend of quality does not vary with scenes (Section 4.11). Optimal parameters may vary slightly across different scenes (*e.g.* different reflectivity), and can be easily obtained by taking one photo of the scene and running the aforementioned simulation.

Adding Color to the Model. Occasionally sensitive information is in the color channel of images, which requires LiShield to distort color for protection. LiShield extends to multi-channel setup by independently generating waveforms for each of the RGB channels, turning white stripes in the previous model into colored ones. To compensate for the intensity loss compared with the white stripes, we need to make the new peak intensity $I'_p = 3I_p$, assuming R, G, and B appear with equal probability.

4.7.2 Circumventing Wild Captures

Based on the foregoing analysis, we identify the following potential holes that attackers can exploit to overcome the striping effect. (i) Manual exposure attack. If an attacker can configure the t_e to satisfy $t_e \approx N t_l$, it can guarantee every row receives almost the same illumination, thus eliminating the stripes during a capture (Figure 4.2(e)). In practice, t_l is unknown to the attacker, but it can try to capture images with different t_e, until seeing a version without obvious stripes[1]. (ii) Multi-frame attack. When the scene is static, an attacker may also combine multiple frames (taking a video and playback) to mitigate the stripes with statistical clues, *e.g.* by averaging or combining rows with maximum intensities from multiple frames. Note that the attacker must keep the camera highly stable, otherwise even pixel-level shift will cause severe deformation when combining multiple frames. (iii) Post-processing attack. Common post-processing techniques (*e.g.*, denoising and de-banding) might be used to repair the corrupted images.

In what follows, we introduce countermeasures to the first two attacks. In Section 4.11.4, we will verify that LiShield's distortion does not fit empirical noise or banding models, so the common post-processing schemes become ineffective.

Frequency Scrambling. To thwart the manual exposure attack, we design a *frequency scrambling*[2] mechanism, which packs multiple waveforms with randomly selected frequencies within each image frame duration. Since the camera exposure time t_e is always fixed within each frame, no single t_e can circumvent all the frequency components.

However, we cannot arbitrarily choose and switch the flickering frequencies for three reasons. *(i)* Multiple frequency values that share a common divisor can satisfy $t_e = Nt_l$ under the same t_e (recall N can be an arbitrary integer). We need to ensure the common divisor is small enough (*i.e.*, least common multiplier of t_l large enough), so that overexposure occurs even for the smallest N. *(ii)* Frequencies should be kept low to maximize image corruption, as evident in Figure 4.3(e) and (f), since the camera's analog gain decreases at high frequencies [203]. *(iii)* Switching between different frequencies may create an additional level of modulation, which will spread the spectrum and generate unexpected low frequency components that become

[1]Digital camera's exposure time cannot be set arbitrarily due to hardware limitation. On Nexus 5, the granularity is around 13μs, *e.g.*, the actual exposure time is 1/1950 when the attacker needs 1/2000. Note that exposure time must be fixed within each frame.

[2]Scrambling and randomization are exchangeable in this chapter.

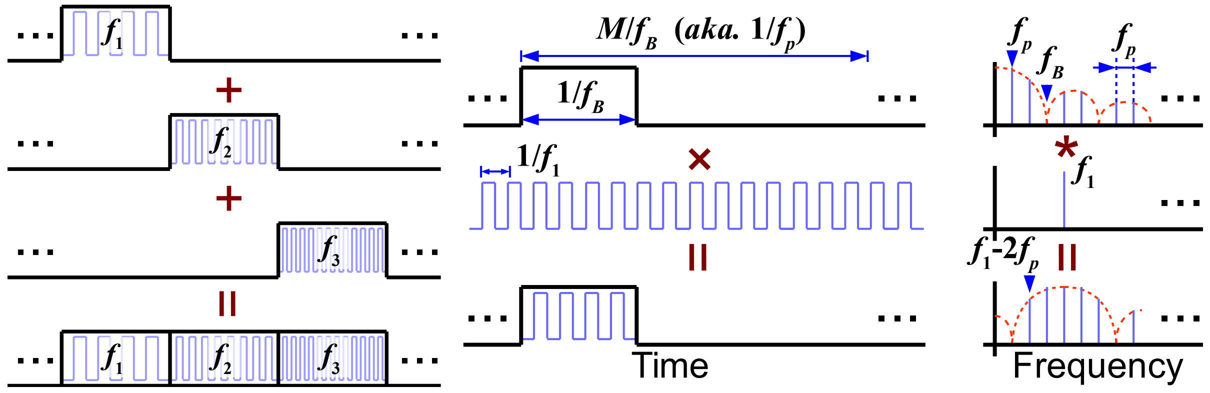


Figure 4.4. Decomposition of frequency randomization waveform and modulation generating side lobes.

perceivable by eyes.

To explore the design space under these constraints, suppose we switch among M frequencies $f_1, f_2, \ldots, f_M$ (in ascending order) at a switching rate f_B. The whole pattern thus repeats itself at rate $f_p = f_B/M$. To pack at least M different frequencies in an image frame, we need $f_B > (M-1)f_r$, or preferably, $f_B > Mf_r$, where f_r is the frame rate, typically around 30 Hz (fps). Note that the switching rate cannot be higher than the lowest scrambling frequency, *i.e.* $f_B \leq f_1$, otherwise the waveforms of f_1 will be truncated. To maximize image corruption, we choose the smallest value for f_1 (*i.e.*, $f_1 = f_B$), and empirically set $f_n = f_B + (n-1)\Delta f, n \in 2, 3, \ldots, M$, where Δf is frequency increment, set to $\Delta f \neq f_B$ to lower the common divisor frequency.

The frequency scrambling can be considered as an M-FSK modulation: essentially, we multiply the waveform corresponding to each frequency with a rectangular wave of frequency f_p and duty cycle $1/M$, which convolves harmonics of the pattern repetition frequency f_p to the spectrum, creating side lobes around each scrambling frequency, spacing f_p apart, as shown in Figure 4.4. These side lobes might appear at low-frequency region and become perceptible by human eyes.

To tackle this challenge, note that for waveforms with frequencies $f_2, f_3, \ldots$, their side lobes are dampened more at lower frequencies compared with f_1, so we only need to focus on

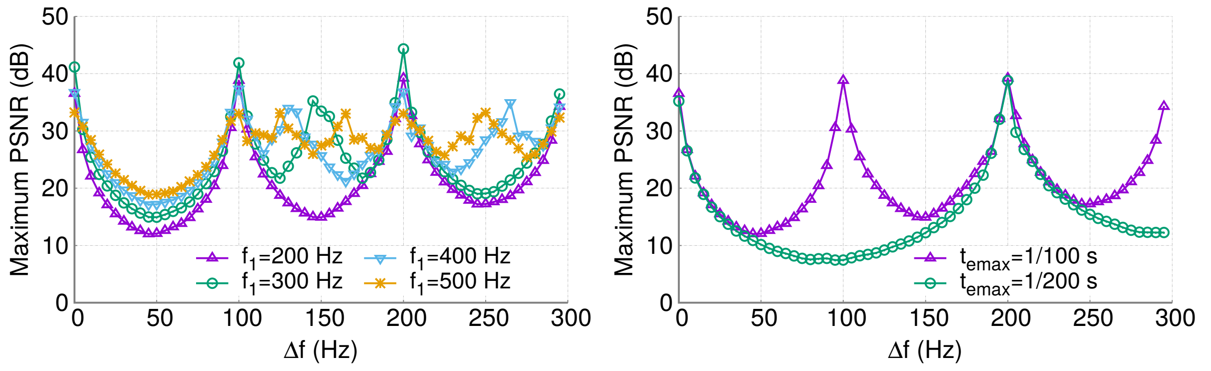


Figure 4.5. Worst-case PSNR with different frequency increment Δf under different parameter settings.

f_1. The side lobes of f_1 are located at $f_1 + kf_p$, where k is an integer. For the side lobe with the lowest frequency, $k = \lfloor f_1/f_p \rfloor$. Since we selected $f_1 = f_B = Mf_p$, the lowest non-DC side lobe is at $f_p = f_B/M$. Therefore, to ensure no side lobe exists below the perceivable threshold $f_{\text{th}} \approx 80$ Hz, we need a small M and large f_B, and hence higher flickering frequency components f_n. Yet increasing the flickering frequencies may weaken LiShield's protection. Fortunately, since LiShield does not require large M (which leads to high f_M) to circumvent the manual exposure attack, the degradation should be tolerable.

To find the optimal Δf and showcase the effectiveness of the frequency scrambling, we choose the case $M = 2$ and $f_1 = f_B$ under 20 kLx peak intensity (to be consistent with our testbed setup in Section 4.10). We then repeat the numerical simulation (Section 4.7.1) to evaluate the attacker's maximum image quality. Figure 4.5(a) shows that the quality has two peaks at $\Delta f = 0$ and 100 Hz, as well as a valley at $\Delta f = 50$ Hz. Note that the positions of these peaks/valleys are independent of f_1 and M, because quality always reaches the maximum at the longest t_e before overexposure happens (denoted as t_{emax} in Figure 4.5(b)). Thus, we set $\Delta f = (1/2)/t_{emax} = 50$ Hz to maximize image disruption. The optimal Δf for other peak intensity settings can be obtained following a similar procedure. Figure 4.5 also shows that, once set to the optimal Δf, frequency randomization can significantly improve LiShield's robustness against manual exposure attacks. Section 4.11 will show more evidence through testbed experiments.

Illumination Intensity Randomization. If attackers repetitively capture a static scene for a sufficiently long duration, they may eventually find at least one clean version for each row across all frames, thus *recovering* the image. LiShield does not guarantee complete protection against such brute-force attacks. However, it can increase the number of frames needed for image recovery, so that the attack becomes infeasible unless the camera can stay perfectly still over a long period of time, during which the attackers may have already been discovered by the owners of the physical space. LiShield achieves the goal by employing illumination intensity randomization, where it randomly switches the magnitude of each ON period across multiple predefined levels, which extends the attacker's search space. We note that the intensity randomization adds another level of modulation, but similar analysis in Section 4.7.2 still applies and can ensure imperceptible operation.

To understand the effectiveness of this scheme, we build a statistical model to estimate the number of frames needed to perfectly recover the image, as if LiShield did not function at all. Suppose the LED waveform has K intensity levels, and the camera has m rows. For simplicity, we assume the intensity levels of each row become uncorrelated after the randomization. Then the probability that one row gets any illumination is $p = t_{\text{on}}/(t_{\text{on}} + t_{\text{off}}) = D_c$. Observe that on average same intensities would reappear approximately every K frames, the possibility of combining L frames to fully recover an image of the static scene is thus:

$$P_{\text{rec}} = \begin{cases} \left[1-(1-D_c)^{L/K}\right]^{m} & \text{(monochrome)} \\ \left[1-(1-D_c)^{L/K}\right]^{3m} & \text{(RGB)} \end{cases} \tag{4.8}$$

Therefore, achieving a given level of P_{rec} becomes increasingly difficult as D_c and K increases, and for higher camera resolution (larger m). For example, to have $P_{\text{rec}} = 90\%$ for $D_c = 0.5$ and $m = 2448$ for 8-mega-pixel cameras, the attacker needs $L = 300$ frames under $K = 10$, and ~ 3000 frames under $K = 100$. For lower duty cycles, recovery becomes even more challenging (*e.g.*, ~ 7000 frames are needed for $D_c = 0.2$, $K = 100$). As we will show later

(Section 4.11), in practice, the attackers cannot keep cameras completely still and all frames aligned at the pixel level, even with a tripod and across a short duration. So Equation 4.8 gives the best performance for such attacks. Note that if the target scene is mobile, then the multi-frame attack becomes impossible, as long as the scene has certain variation across K frames. The effectiveness of intensity randomization will be further justified in our testbed experiments (Section 4.11).

4.8 Scene Recovery using Computational Shutters

To allow authorized users to capture the scene while maintaining protection against unauthorized attackers, we need to impose additional constraints on the LED waveform. LiShield's solution leverages a secure side channel (*e.g.* visible light communication [24] or Wi-Fi) between authorized users and the smart LED, which conveys secret information such as frame timing and waveform parameters[3].

A naive solution is to stop flickering when authorized users are recording. However, since attackers may be co-located with the authorized users, this enables them to capture one or more frames that have part of the clean scene, which compromises privacy and security. To counteract such cases, we design unique waveforms for the LED to minimize the flicker-free duration.

4.8.1 Dynamic Scene Video Restoration

To authorize a camera to capture a dynamic scene, each individual frame within the video must be recoverable. To achieve this, the authorized camera needs to convey its exposure time setting t_e^u to the smart LED via the secure side channel, and synchronize its clock (for controlling capturing time) with the smart LED's clock (for controlling the waveform), so the smart LED can send recoverable waveforms precisely during the capture of the authorized

[3]Such information can be protected by existing encryption algorithms and systems, which are already mature and thus beyond the scope of this project.

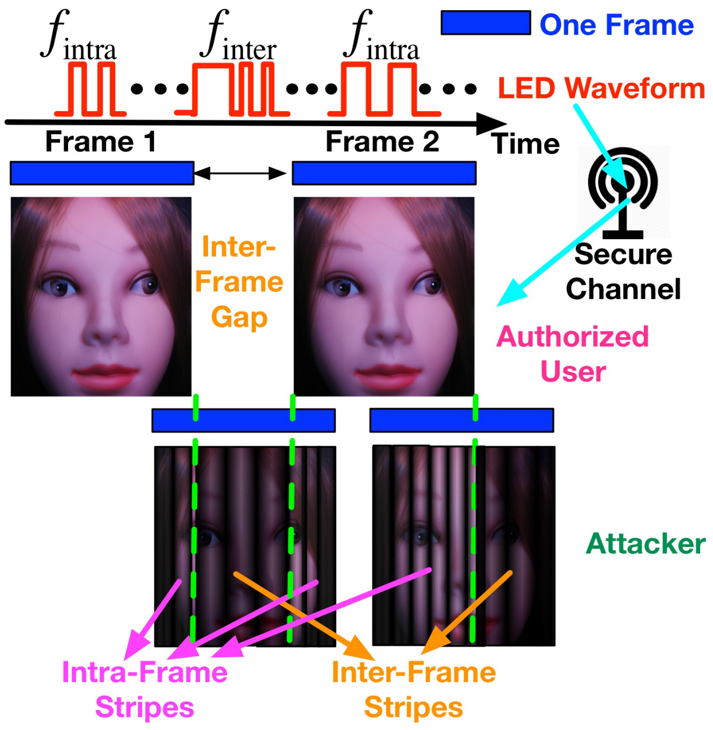


Figure 4.6. Enabling authorized users to capture dynamic scenes while corrupting unauthorized users.

camera. State-of-the-art time synchronization mechanisms through visible light [99] or wireless side-channels [149, 37, 143] can already achieve μs of accuracy, sufficient to synchronize the LiShield smart LED with the camera at a resolution that is finer than the rolling shutter period (typically tens of μs).

Recall that the camera can evade the striping effects if $t_e = Nt_l$ (phase does not matter, see Section 4.7.2). So to authorize the user with exposure t_e^u, LiShield simply needs to set its flickering frequency $f_a = 1/t_l = N/t_e^u$ ($N = 1, 2, \ldots$) and maintain its peak intensity within each frame. In addition, the t_e^u and corresponding flickering frequency f_a can be varied on a frame by frame basis, making it impossible for an attacker to resolve the correct exposure time by trial-and-error (Section 4.7.2).

Meanwhile, when the authorized camera is not recording at its maximum possible rate (*e.g.*, a 30 fps camera recording at 25 fps), there will be an interval (*i.e.*, inter-frame gap) where

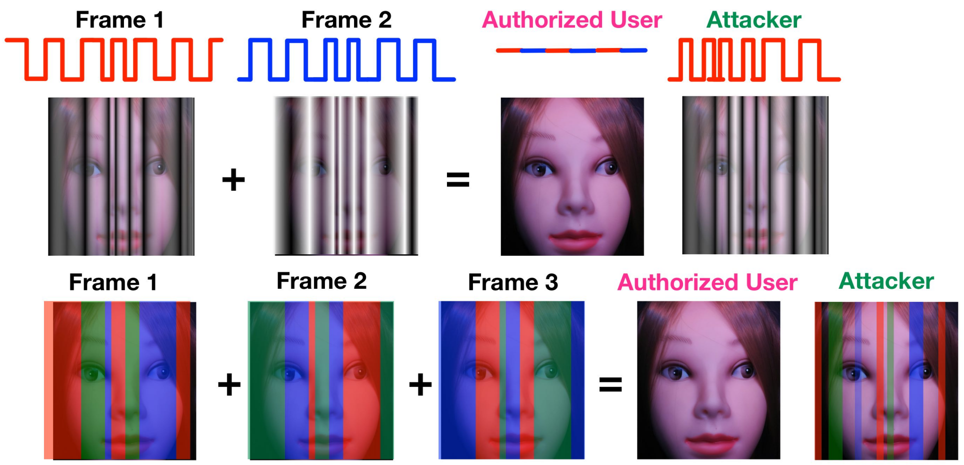


Figure 4.7. The impact of multi-frame recovery on authorized user and attacker, respectively.

the camera pauses capturing. LiShield packs random flickering frequencies other than f_a into the inter-frame gap, so as to achieve the same scrambling effect as described in Section 4.7.2, without compromising the authorized capturing. Figure 4.6 depicts one example, where f_{intra} and f_{inter} denote intra-frame and inter-frame frequencies, respectively.

4.8.2 Static Scene Image Recovery

When the target scene is static, the authorized user may capture a few complementary frames at a specific time to recover the scene as depicted in Figure 4.7, where frequency and intensity randomization (Section 4.7.2) are employed in each frame to ensure robustness. While it does require recording a very short video, the process is extremely short (200ms at most) and barely noticeable to the authorized user. Meanwhile, an out-of-sync attacker will still receive corrupted images that cannot reconstruct the original scene even after combined.

Suppose a static scene is to be recovered using L_f frames, referred to as *critical frames*. To prevent attackers from launching the multi-frame attack, the timing of the critical frames is negotiated only between the smart LED and the authorized user through the secure side channel. These L_f frames together must contain the information of the entire scene, *i.e.* they must be complementary, as shown in Figure 4.7. Meanwhile, all other frames will follow the normal flickering pattern as discussed in Section 4.7. Since the attackers cannot identify nor predict the timing of the critical frames , the best they can do is to launch the brute-force multi-frame attack, which has been discussed in Section 4.7.2.

4.9 Light Encoding for Appearance Watermarking

High-intensity ambient light sources (*e.g.* sunlight, legacy lighting, flashlights) can create strong interference to LiShield's illumination waveform, degrading the contrast by adding a constant intensity to both the bright and dark stripes, which may weaken LiShield's protection. In such scenarios, LiShield degrades itself to a *barcode mode*, where it embeds barcode in the physical scene to convey privacy policies. The barcode forms low-contrast stripes, which may not thoroughly corrupt the images of the scene, but can still be detected by online photo-distributing hubs (*e.g.*, social website servers) who automatically enforce the policies, without the cooperation of the uploader or evidence visible by naked eye. LiShield forms the watermark with just a single light fixture, instead of active displays (*e.g.*, projectors) that are required by conventional systems. The key challenge here is: how should LiShield encode the information, so that it can be robustly conveyed to the policy enforcers, despite the (uncontrollable) attacker camera settings? We now describe how LiShield's barcode design meets the challenge.

4.9.1 Illumination Effect Embedding

LiShield's barcode packs multiple frequencies in every image (or in every frame of a video) following Section 4.7.2, but aims to map the *ratios between frequencies* into digital information. Suppose LiShield embeds two waveforms with frequencies F_0 and F_1, it chooses the two frequency components such that F_1/F_0 equals to a value R_p well known to the policy enforcers. In other words, the presence of R_p conveys "no distribution/sharing allowed". This encoding mechanism is robust against camera settings [4].

Since physical scenes usually comprise a mix of spatial frequencies, and spectral power

[4]Although width of stripes is affected by sampling interval t_s and exposure time t_e (Figure 4.2(a) and (b)), ratio of stripe widths resulted from two frequencies (which equals to R_p) remains constant.

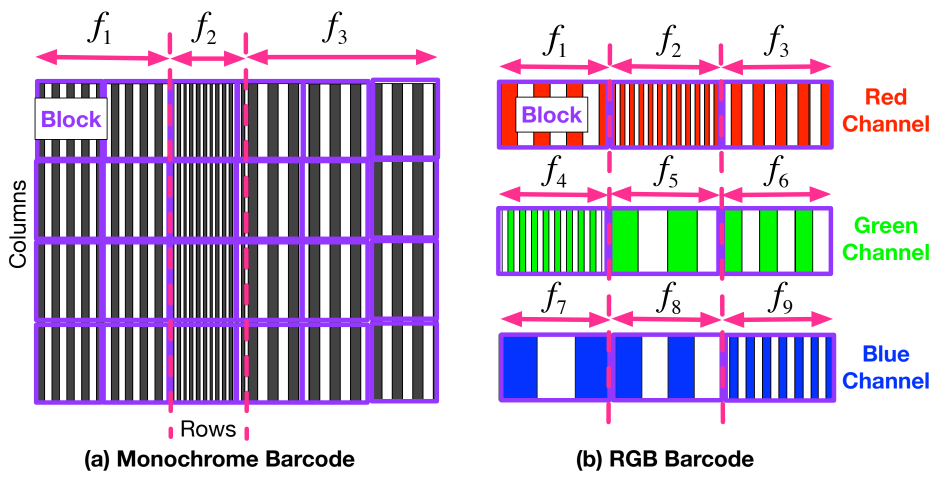


Figure 4.8. Barcode design for monochrome and RGB LED.

rolls off in higher spatial frequencies thanks to camera lenses limited bandwidth [51] while temporal frequencies are unaffected, LiShield's barcode uses frequencies that are much higher than the natural frequencies ($>$ 400Hz) in the scene to avoid interference. It is worth noting that since the rolling-shutter sampling rate of all cameras falls in a range (30 kHz to slightly over 100 kHz [203]), LiShield limits its highest flickering frequency to 15 kHz, which respects the Nyquist sampling theorem so that the barcode can eventually be recovered without any aliasing effect.

To further improve robustness, LiShield leverages redundancy. It embeds multiple pairs of frequency components to make multiple values of R_p. In this way, LiShield can pack different R_p either at different rows of the image or in different color channels, further mitigating interference caused by intrinsic spatial patterns within the scene. Figure 4.8 illustrates an example of monochrome ($C_3^2 = 3$ R_p values) and RGB LEDs ($C_{3\times3}^2 = 36$ R_p values). Note that the same mechanism can be used to increase the amount of information in the barcode, but this is beyond the scope of the present work.

ALGORITHM 2: Barcode Detection

```
Input: image $I$ $(m \times n)$, $n_{b_r}$, $n_{b_c}$, $T_b$, $S_b$, $M_b$, $m_b = 0$, $F = \varnothing$, $D_p = \varnothing$, $B = \varnothing$, $f_s = 30$ kHz
Output: whether $I$ is protected
crop $I$ to $n_{b_r} \times n_{b_c}$-size blocks, store in set $B$;
for $b \in B$ do
    $s_r \leftarrow n_{b_r} \times 1 \leftarrow \text{mean}(n_{b_r} \times n_{b_c})$;
    $F_D \leftarrow \text{detrend}(FFT(s_r, f_s))$;
    pick $M_p$ maximum peaks $F_p \in F_D$,
    400 Hz$\leqslant F_p \leqslant$15 kHz;
    $F \leftarrow F \cup F_p$;
end
$D_p \leftarrow D_p \cup (f_i/f_j)$, $\forall f_i, f_j \in F$;
for $d_p \in D_p$ do
    if $d_p \in [R_p - T_b, R_p + T_b]$, $\forall R_p \in S_b$ then
        $m_b \leftarrow m_b + 1$;
    end
end
if $m_b \geq M_b$ then
    $I$ is protected;
end
```

4.9.2 Pattern Detection and Recognition

Since the barcode contains M frequencies, *i.e.* $f_n = f_B + (n-1)\Delta f, n \in 2, 3, \ldots, M$ (Section 4.7.2), there are $M_R = C_M^2$ possible frequency ratio values across the image for monochrome barcode ($M_R = C_{M\times 3}^2$ for RGB barcode). Δf must be set large enough to avoid confusion ($\Delta f = 200$ Hz in experiments). The barcode decoder, running on the policy enforcer, recognizes the image as protected if there are at least M_b values that roughly match the known ratio R_p, *i.e.*, when the value falls within T_b of R_p. We empirically set $M_b = \lceil \gamma M_R + M_{\text{att}} \rceil$ where M_{att} is number of R_p removed by manual exposure attack (Section 4.7.2). γ and T_b are determined by bounding the false positive rate following an empirical procedure (to be discussed in Section 4.11.3).

To detect the frequency ratios, LiShield first partitions the image into $n_{b_r} \times n_{b_c}$ blocks, across both rows and columns, either within the monochrome channel or among all 3 RGB channels. Dividing image by columns provides LiShield multiple copies of the same frequency block, in case some of them are interfered by spatial patterns in the scene. For example, in Figure 4.8, $n_{b_r} = 6$ and $n_{b_c} = 4$. For each block, LiShield averages the intensity of each row to get a one-dimension time series s_r of length $L_f = m/n_{b_r}$, given total m rows on the image. LiShield then runs FFT over each series to extract the M_p strongest frequencies. Note that LiShield's detector allows more than one frequencies to appear in one block. Finally, LiShield combines all unique frequencies extracted from each block and computes all frequency ratios (within and across color channels in the case of RGB barcode). Algorithm 2 describes the procedure of barcode detection.

LiShield's redundancy in barcode ensures that the barcode cannot be completely destroyed, unless nearly all frequencies are distorted by processing the image, which will in turn cause strong deformation on the scene. We will verify the robustness of this scheme through testbed experiments (Section 4.11.4).

4.10 Implementation and Hardware Design

Testbed Setup. Figure 4.9 shows our smart LED prototype, and the target scenes containing 5 capture-sensitive objects (document and painting are 2-D objects and others are all 3-D objects). We mount the LED inside a diffusive plastic cover similar to conventional ceiling light covers. We use a programmable motor [20] to hold the camera and control its distance/orientation, in order to create static or dynamic scene setup in a repeatable manner.

Smart LED Modules. Commercial-of-the-shelf (COTS) household LED bulbs rely on integrated drivers to regulate LED's current [168, 103]. A dimming input is usually available on these drivers for controlling the current dynamically. We build our smart bulb based on the same topology as these COTS LED bulbs. For safety, we use 19V DC laptop power supplies instead of wall AC power, and NCL30160 [120] LED drivers which allow dimming at nearly 100 kHz with arbitrary OOK waveform. The intelligent bulb has built-in independent RGB/white channels for controlling color/intensity. Each channel can be controlled by a separate waveform, with 4 LED chips in series, at a driving current of 800 mA. In total, the 3 channels consume approximately 25 W peak power, close to ordinary office LED troffer fixtures. However, since LiShield's OOK

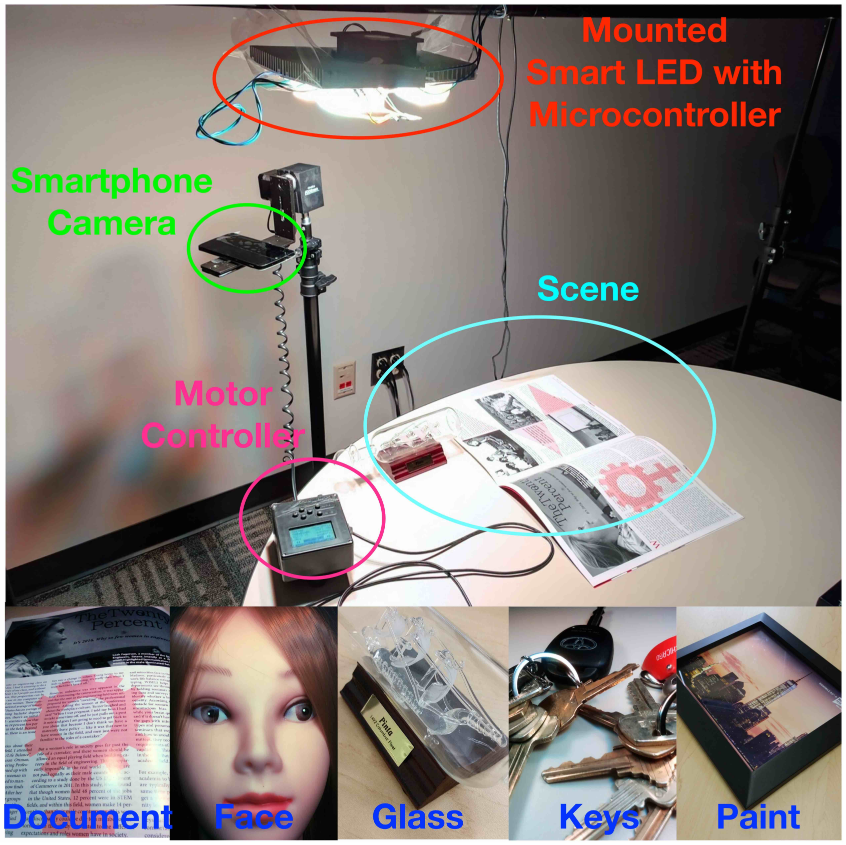


Figure 4.9. Experimental setup and multiple scenes we used.

waveform has a duty cycle much lower than 1 (Section 4.7), the actual perceptible brightness is significantly lower. As a result, multiple LED modules can be used to improve light intensity. Figure 4.10 depicts the circuit for each color channel and shows a photo of the whole module.

The dimming input signals of each channel are controlled by an STM32 [156] microcontroller unit (MCU), which generates the OOK waveform as specified by LiShield. For flexible reconfiguration, we generate digitized waveforms in MATLAB on a laptop or Android app on a smartphone instead, which are then passed to the MCU via USB.

Android Program for Normal, Authorized and Attacking Cameras. Unless otherwise noted, we use Nexus 5 [32] with stock ROM as our benchmark device. We assume that normal users use the stock camera app with default settings (including auto-exposure), while a malicious attacker can manually tune the camera parameters (*e.g.*, using the Open Camera app

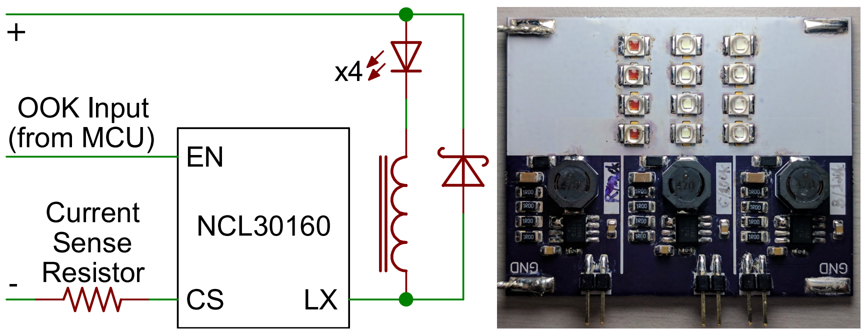
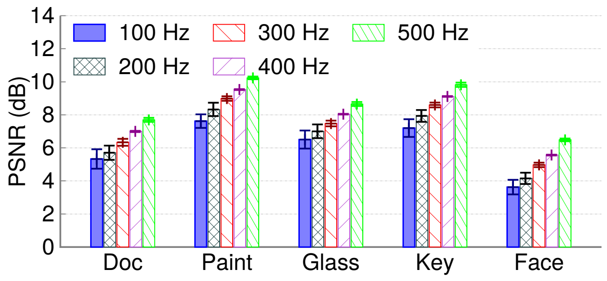


Figure 4.10. Simplified circuit diagram and photo for the smart LED module.

[64]). By default, the camera ISO is set to the lowest value (100) since it is the most beneficial for attackers, as it allows longer exposure to smooth out the stripes without causing overexposure. To implement the authorization mechanism (Section 4.8), we develop a specialized app for the authorized smartphone, which uses Android's Camera2 API [52] to precisely control the exposure time, as well as communicate with the smart LED's MCU via USB. Since current Android camera APIs do not support precise frame timing, the app requests the smart LED to synchronize with the camera by altering its waveform.

Image Processing Attack. We have implemented the attacking algorithms in Section 4.7.2, which are specifically designed to combine/process the captured image, aiming to eliminate LiShield's stripe distortion. In addition, we implement classical image processing techniques, including denoising and debanding, which may be attempted by attackers. For denoising, we use the Haar-wavelet thresholding [22], non-local-means (NLmeans) [12] and BM3D [23], which are among the most popular algorithms [144]. For Haar-wavelet and NLmeans, we use the G'MIC [72] plugin of the GIMP [162] image processing program. For BM3D, we use a CUDA implementation [66] since it is significantly faster and practical than CPU-base implementations. As for debanding, we use the Banding Denoise [36] and Unstrip [11] in the G'MIC [72] plugin.

Metrics. Small displacement and vibration of the camera are inevitable in physical environment, which is known to affect the SSIM of captured images significantly [184]. Thus, we quantify the image quality degradation with the enhanced CW-SSIM [139], which is insensitive under such translations but similar to SSIM otherwise. Since PSNR shows similar trends with



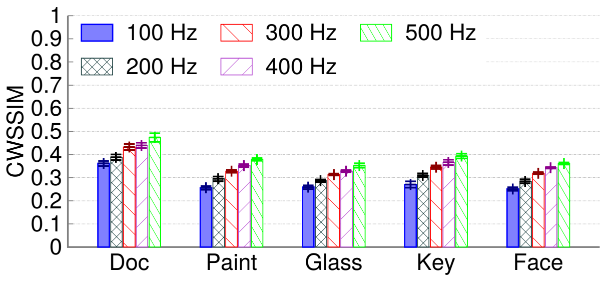


Figure 4.11. Impact of flickering frequency on quality.

SSIM, we omit it in the experiments except for a few cases. Besides, we employ the CIEDE2000 [102] to compute the degradation of the image color quality when the RGB LED is used.

4.11 Prototype Testing and Evaluation

4.11.1 Efficacy of Stripe Pattern on Images

Impact of Flickering Frequency. We first verify LiShield's basic protection scheme (Section 4.7) with 5 static scenes, monochrome LEDs, and OOK waveform without frequency randomization, while the attacker's camera uses automatic exposure. Without LiShield, the measured image quality stays high, with PSNR$>$ 30 dB and CW-SSIM$>$ 0.9 (slightly lower than simulation results due to digital noises in real cameras). Despite the use of a basic configuration, LiShield degrades the image quality by 3 to 10 dB for PSNR and 0.25 to 0.45 for CW-SSIM (Figure 4.11). We notice that the quality worsens slightly as flickering frequency decreases from 500 Hz to 100 Hz, as the image sensor has higher analog gain at lower flickering frequencies [203]. In addition, different scenes suffer from different levels of disruption, depending on the scene's structure and reflection rate. As a visual quality benchmark, Figure 4.12 plots the same scene with different qualities under flickering.

Impact of Waveform Duty Cycle. We use 100 Hz flickering frequency on the document scene as a representative setup [5] to study the impact of duty cycle of the emitted waveform. Here we enable automatic exposure to study the stripes' impact alone. Figure 4.13 shows that lowering

[5]Unless otherwise noted, the rest of experiments use the same setup.

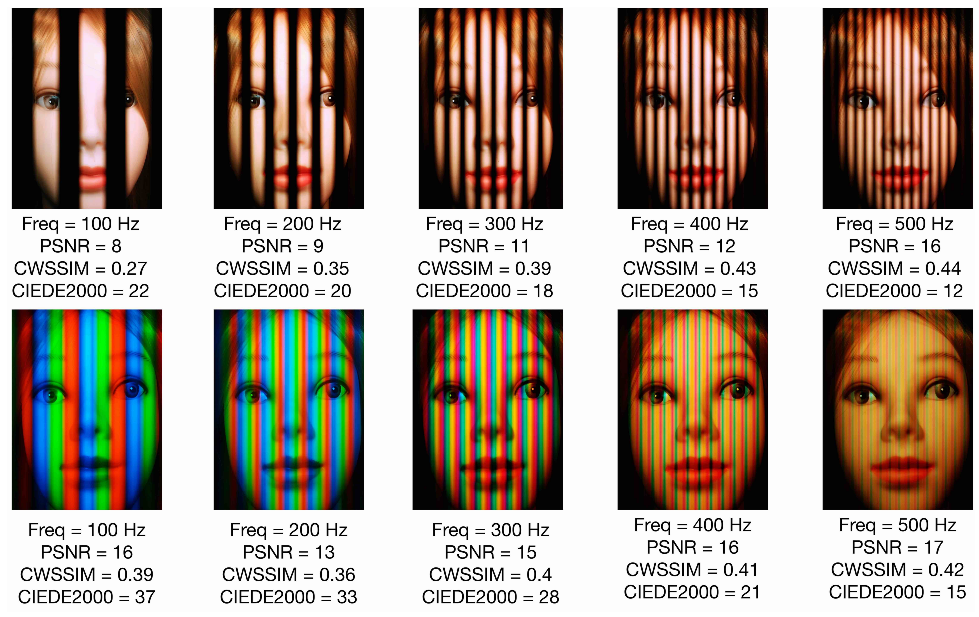


Figure 4.12. Image quality levels on a benchmark image.

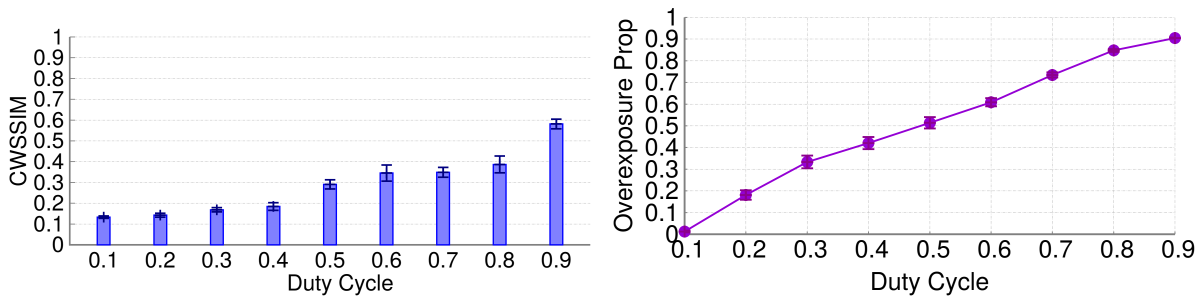
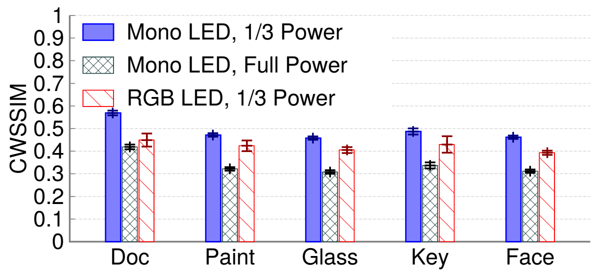


Figure 4.13. *Left:* Impact of duty cycle on quality with automatic exposure. *Right:* Impact of duty cycle on overexposure area with fixed exposure.

the duty cycle from 0.9 to 0.1 degrades the image quality dramatically, with CW-SSIM from nearly 0.6 to just over 0.1. However, a higher duty cycle leads to more light influx and larger overexposure area (Figure 4.13) when fixed exposure is used by the attacker (here $t_e = 1/200$ s). To leverage both types of quality degradations (*i.e.*, flickering stripes and overexposure), the LED should adopt a relatively moderate duty cycle but high peak intensity, which echoes our model in Section 4.7.1.

Impact of RGB Color Distortion. We further verify the color-distortion impact when the RGB flickering is turned on. The results (Figure 4.14) demonstrate slightly weaker quality degradation when its peak intensity is the same as monochrome LED (and thus average intensity



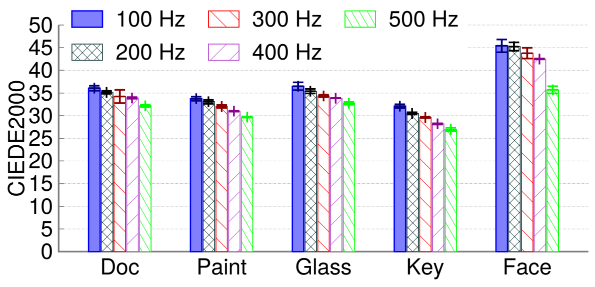
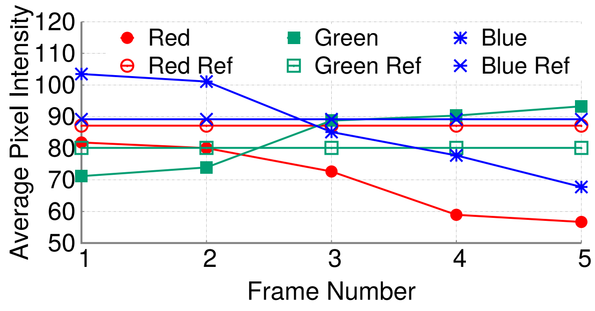


Figure 4.14. *Left:* Impact of color on quality. *Right:* CIEDE2000 for measuring color distortion.



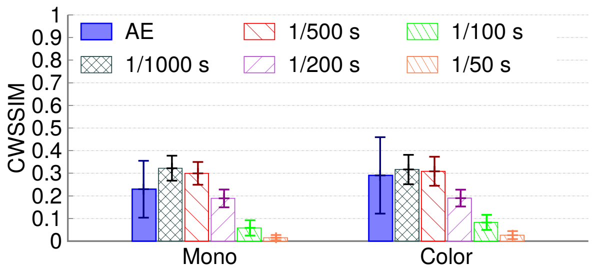


Figure 4.15. *Left:* Impact on automatic white balance. *Right:* Impact on dynamic scene.

is only 1/3). But the quality degradation is stronger if the RGB LED has the same average intensity as monochrome LED. Besides, the color distortion makes an additional independent impact. The corresponding CIEDE2000 metric (Figure 4.14) escalates up to 45, way beyond the human-tolerable threshold 6 [102]. This implies the scene is no longer considered viewable by average viewers.

Two *bonus effects* from our RGB LED are observed: *(i)* The structural distortion from the stripes disrupts the camera's auto-focus function, often making the captured scene extremely blurred. This is because under LiShield, the contrast of bands no longer depends on focusing accuracy, which breaks the assumption of the auto-focus mechanism. *(ii)* The color bands also mislead the automatic white balance function across all 5 different scenes, since the camera can no longer identify a clean region in the image to calibrate itself and thus hesitates, as shown in Figure 4.15.

Impact on Dynamic Scenes. To create a dynamic scene, we use the motor to rotate the smartphone, creating relative motion at three different speeds (45, 100, and 145 degrees/second).

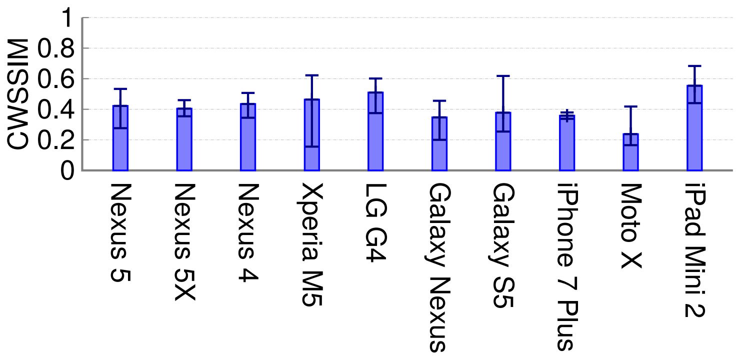


Figure 4.16. Impact of device heterogeneity. Error bars show *std.* across OOK waveforms with different frequencies (100 Hz to 500 Hz).

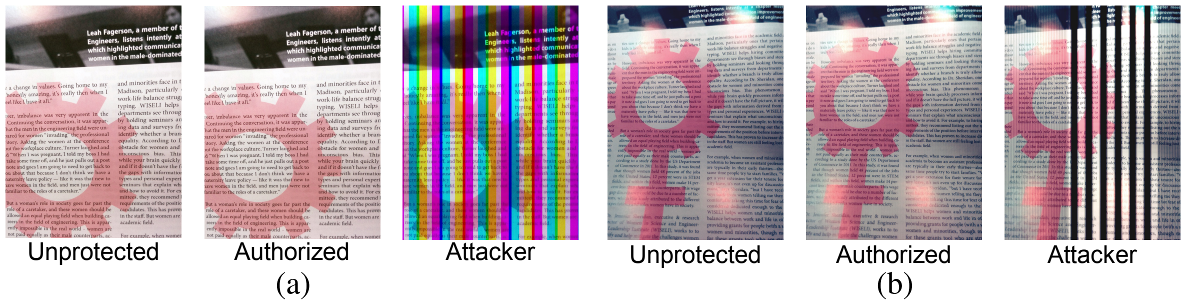


Figure 4.17. Frames observed by authorized users and attackers for (a) static scene and (b) dynamic scene.

Figure 4.15 shows the average quality among all 3 speeds, which indicates that dynamic scene experiences worse quality under LiShield due to motion blur. Moreover, if the exposure time is larger than 1/100 s, then overexposure and motion blur together further reduce the quality (PSNR < 6, CW-SSIM < 0.1). Thus, dynamic objects further decrease the adjustment range of exposure time and make manual exposure attacks more ineffective.

Impact of Device Heterogeneity. We cross-validate the impact of LiShield on 10 common smartphone cameras. Figure 4.16 shows the image quality varies slightly, due to different sampling rates across devices resulting in stripes of different widths. However, the quality remains at an intolerably low level across devices. Thus LiShield's protection mechanism works across typical smartphone camera models.

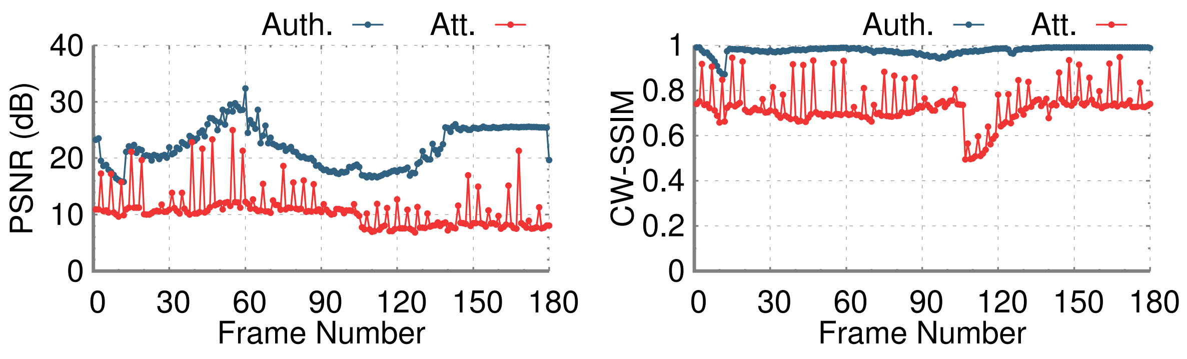


Figure 4.18. Video quality with and without authorization.

4.11.2 Restoring Appearances with Certified Cameras

We developed an app (Section 4.10) that allows a user to capture critical frames on static scene protected by our RGB LED, and then recover the scene following Section 4.8. The resulting image quality (PSNR = 25dB, CW-SSIM = 0.9, CIEDE2000 = 5) is comparable to the ideal setting when we disable LiShield's LED modulation (Figure 4.17 shows example frames extracted from a recorded video). In contrast, the attacker suffers intolerable image corruption (PSNR = 13dB, CW-SSIM = 0.56, CIEDE2000 = 34) by combining same number of randomly selected frames (Section 4.7.2).

For the dynamic scene, we set $f_{\text{intra}} = 1$ kHz and $f_{\text{inter}} = 300$ Hz (Section 4.8.1). From Figure 4.18, we can see the authorized user has much higher quality (PSNR=25dB, CW-SSIM=0.98 in average) compared with attacker (PSNR = 10dB, CW-SSIM = 0.6 in average). This can be seen by resulting image frames in Figure 4.17 where attacker suffers from both intra-frame and inter-frame stripes. Thus LiShield's authorization scheme is effective in unblocking specific users while maintaining protection against attackers.

4.11.3 Placing and Uncovering Hidden Barcodes

We first determine general optimal parameters for LiShield's barcode detector (γ, T_b, n_{b_r} and n_{b_c} in Section 4.9), based on the following metrics. *(i)* False alarm rate. We run the detector on 200 images (random real-world scenes) in the SIPI database [186], and measure

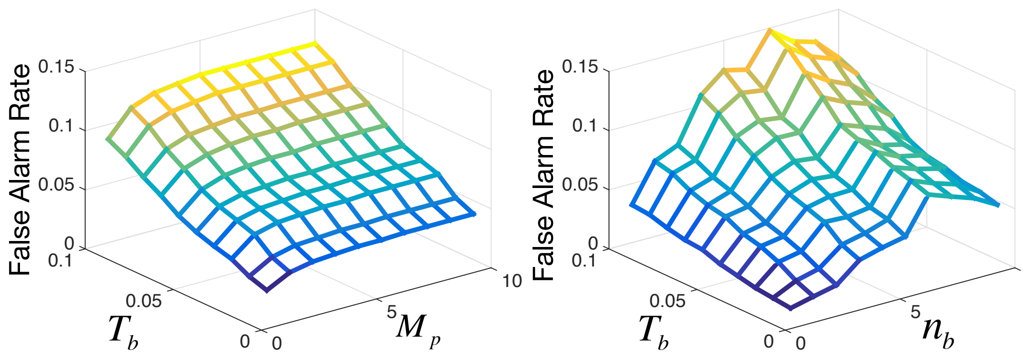


Figure 4.19. False alarm ratio across detector settings.

the probability that a barcode is detected from clean image. *(ii)* Detection rate. We embed monochrome barcodes with different f_1 from 400 Hz to 10 kHz with 200 Hz switching frequency. For each f_1, we embed 3 frequencies (*i.e.*, $M_R = 3$ in Section 4.9) with $\Delta f = 200$ Hz interval and capture 300 images with these barcodes over a benchmark scene (without loss of generality) to obtain detection rate. For simplicity we set $n_b = n_{b_r} = n_{b_c}$. Figure 4.19 plots the fraction of falsely detected frequency ratios (*i.e.*, R_p in Section 4.9) over total number of ratios, while Figure 4.20 shows successful detection rate under the same set of parameters. Considering the trade-off between false alarm and detection, we choose $T_b = 0.05$, $M_p = 2$ and $n_b = 4$ to bound the false alarm rate below 5%, and set $M_b = \lceil 2 \times 5\% \times M_R + M_{\text{att}} \rceil = 3$ to guarantee no false alarm for barcodes with 3 frequencies ($M_{\text{att}} = 2$ since manual exposure attack can remove two R_p), while ensuring around 90% detection rate for monochrome barcode.

Using the above configuration and frequencies in Tables 4.1 and 4.2, Figure 4.21 shows that detection rate for RGB barcodes is close to 100% with or without manual exposure attack, while being slightly below 90% for monochrome barcodes if attacked. We conclude that LiShield's barcode detector provides reliable detection, while RGB barcodes are more detectable and robust than monochrome ones, thanks to extra redundancy provided by color channels.

An attacker may post-process the image in an attempt to remove the watermark. However, thanks to the redundancy of the barcode, the attacker will have to deform most parts of the image,

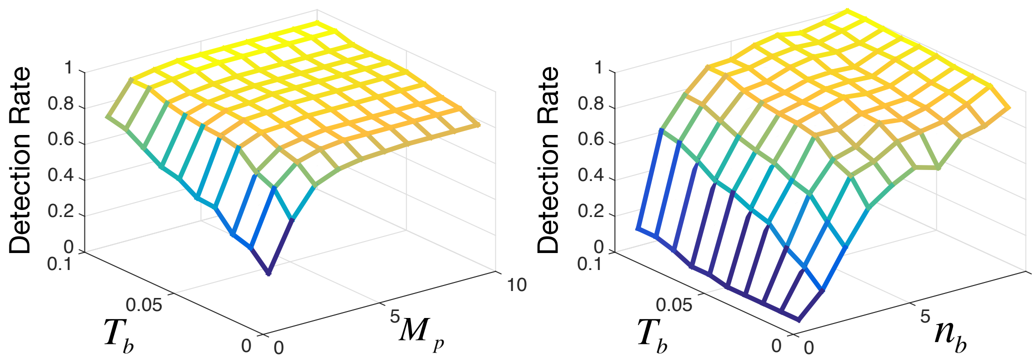


Figure 4.20. Detection rate across detector settings.

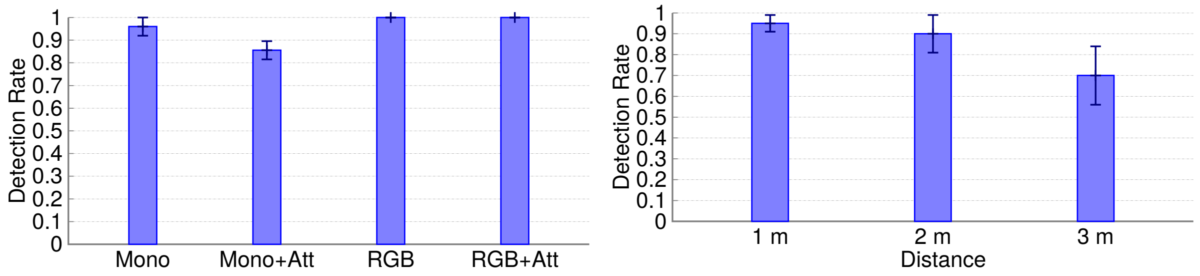


Figure 4.21. *Left:* Detection rate of monochrome and RGB barcode design. *Right:* Barcode detection rate across distance under a single LED.

Table 4.1. Flicker-free configurations for monochrome barcode.

Seq	f_1 (Hz)	f_2 (Hz)	f_3 (Hz)	t_{att} (s)
1	400	600	800	1/400, 1/600, 1/800
2	1000	1200	1400	1/1000, 1/1200, 1/1400
3	1600	1800	2000	1/1600, 1/1800, 1/2000

which greatly reduces the image quality and makes the attack nonviable.

4.11.4 Robustness Against Camera Maneuvers

Manual Exposure Attack. One possible attack against LiShield is to manually set the exposure time t_e to smooth out the flickering patterns (Section 4.7.2). Figure 4.22 shows that although the image quality first increases with t_e, it drops sharply as overexposure occurs. Therefore, LiShield traps the attacker in either extreme by optimizing the waveform (Section 4.7.1), and thwarts any attempts through exposure time configuration.

We further test the effectiveness of randomization as configured in Table 4.3 with auto-

Table 4.2. Flicker-free configurations for RGB barcode.

Color	f_1 (Hz)	f_2 (Hz)	f_3 (Hz)	t_{att} (s)
Red	400	600	800	$1/2000 \sim 1/400$
Green	1000	1200	1400	$1/2000 \sim 1/400$
Blue	1600	1800	2000	$1/2000 \sim 1/400$

Table 4.3. Flicker-free configurations for frequency randomization. f_c, t_{att} represent center frequency and attacker's exposure time.

M	$f_B = f_1$ (Hz)	Δf (Hz)	f_c (Hz)	t_{att} (s)
2	200	50	225	1/225
3	300	50	350	1/350
4	400	50	475	1/475
5	500	50	600	1/600
6	600	50	725	1/725

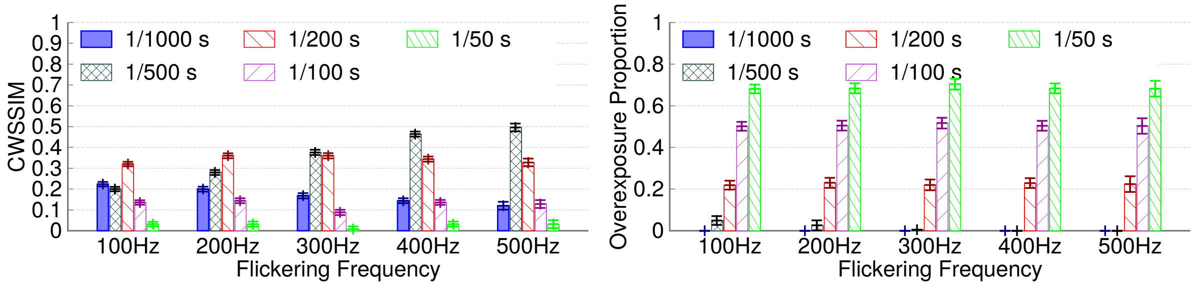


Figure 4.22. Quality and overexposed proportion with fixed-exposure camera.

matic exposure (except for attacker). Figure 4.23 shows that the image quality with scrambling is comparable with a single frequency of f_1 and f_c, thus frequency randomization does not cause much difference in image quality. Note that the image quality varies only slightly with the number of frequencies, implying it is insensitive to LiShield's frequency randomization pattern. We assume the exposure time is $t_{\text{att}} = 1/f_c$, which is optimistic for the attacker. Results show that image quality does not vary significantly (compared with attacks to stripes without randomization), showing LiShield's robustness against such attacks.

Multi-Frame Attack. Figure 4.24 plots the recovered scene's quality under the multi-frame attack. Here we set t_e to be 1/500 s to avoid overexposure and then record a video in 30 fps. When a tripod is used, PSNR goes to 30 dB but CW-SSIM remains low at 0.5 using 1000 frames, which means the impact of stripes on the structure of scenes is still strong although intensity does

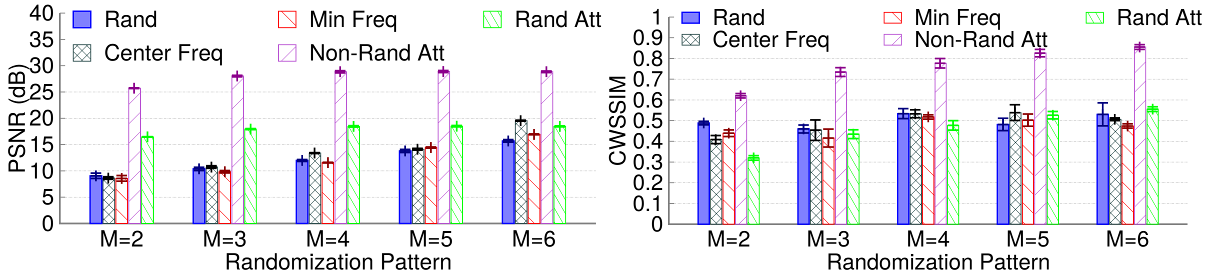


Figure 4.23. Quality with and without frequency randomization.

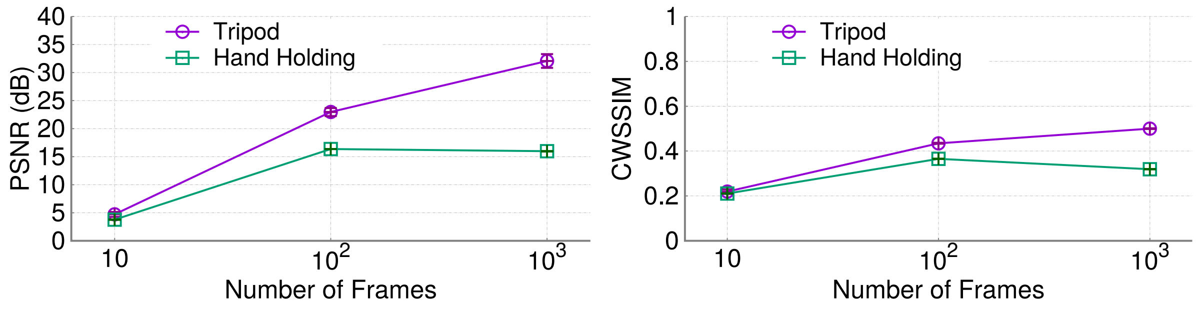


Figure 4.24. Image quality with number of frames of multi-frame attack.

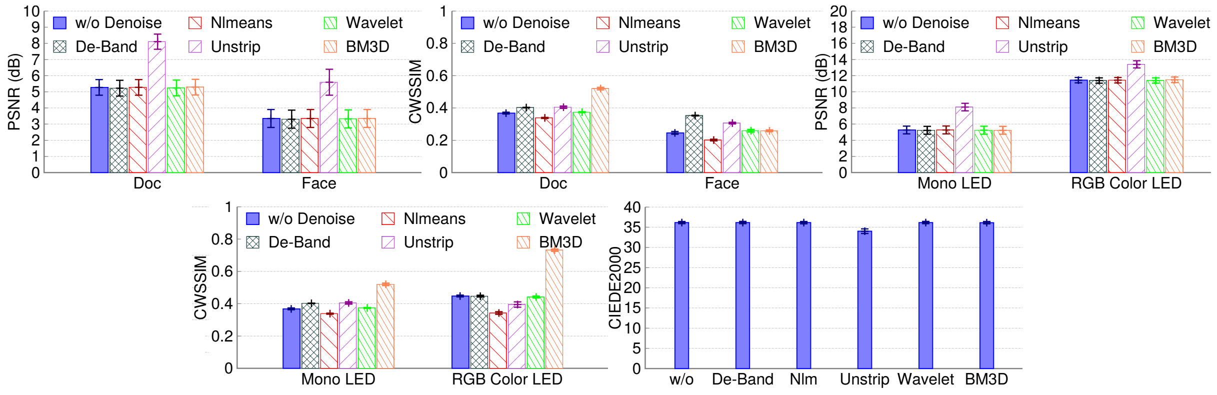


Figure 4.25. Effects of denoising and de-banding image processing algorithms.

not differ substantially, making quality still unacceptable for professionals who spend such a significant cost. We also ask 5 volunteers to hold the smartphone as stable as they can on a table, and Figure 4.24 shows the quality is even lower because it is impossible to completely avoid dithering with hands even with anti-shake technology, making recovered scenes unviewable. Extending the recording duration increases the number of frames recorded by the attacker, but it also increases disturbance and probability of being identified by the protected user, making it risky and impractical for the attacker to pursue higher quality.

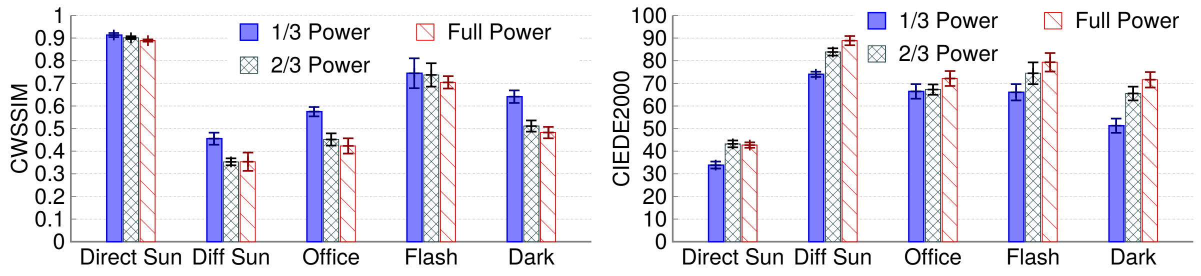


Figure 4.26. Image quality under ambient lights.

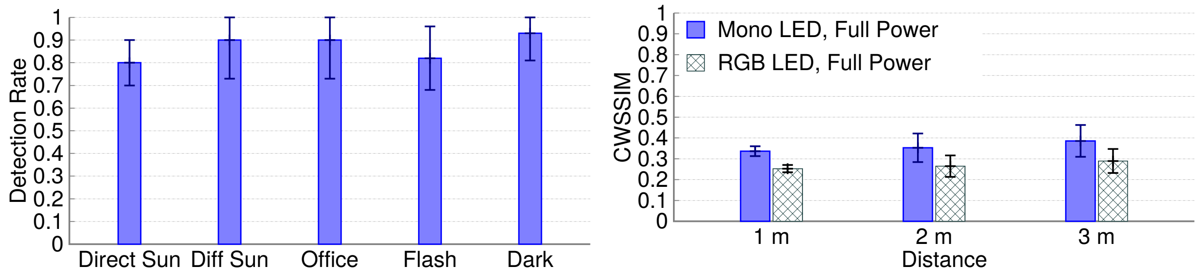


Figure 4.27. *Left:* Barcode detection rate with ambient light intensity. *Right:* Image quality with different distances under a single LED.

Image Recovery Processing Attack. Figure 4.25 shows the image quality after post-processing with denoising or de-banding (Section 4.10). The denoising methods fail to improve the quality significantly as the disruption pattern of LiShield does not fit any known noise model (*e.g.* the commonly used Gaussian model). BM3D can improve the CW-SSIM slightly because it decreases contrast slightly, but the PSNR remains low. The deformation removal methods (*i.e.*, de-banding and unstriping) do not help either, since the interpolation process cannot bring back the correct pixel values. The CIEDE2000 color metric also shows a low quality (well above 6). Thus, it is practically impossible to remove LiShield's impact by image processing, despite some unnoticeable increase of the image quality. More advanced computer vision techniques may provide better recovery, but even they will not recover the *exactly original scene* since information is already lost at capture time.

Impact of Ambient Light. We evaluate LiShield's performance under different types of ambient lights and LED power to verify LiShield's robustness. As shown in Figure 4.26, the stripes are almost entirely removed under direct sunlight due to its extremely high intensity, making the quality comparable with the unprotected case (PSNR>30dB, CW-SSIM>0.9). However, CIEDE2000 remains relatively high as LED's light affects the scene's color tone significantly, which explains unexpected image quality degradation under diffused sunlight and office light in Figure 4.26. The flashlight can increase the quality slightly thanks to its close distance to the scene, but the improvement is marginal and far from unprotected. In addition, Figure 4.27 shows a slight decrease in the detection rate of barcodes under direct sunlight, but the decrease is marginal in every case. Thus, we conclude that LiShield is robust against ambient light, and still guarantees protection with barcode under direct sunlight.

Impact of Distance. We vary the distance between the camera and a single LED from 1 m to 3 m. The scene resolution lowers at a longer distance. Figure 4.21 shows the barcode detection rate remains high ($> 90\%$) at 2 m range (where the bright area is only 1/4 on the image compared with the 1 m case). However, only a 70% rate can be achieved at the 3 m range since the bright area is too small on the image (1/9). However, multiple LEDs may be distributed to increase the coverage. To make a fair comparison on quality, we tailor the same scene from the image to avoid interference from surrounding objects. Figure 4.27 shows that even under 3m, CW-SSIM is still way below 0.9 and the quality only increases slightly with distance. Thus, LiShield's working range can cover most of the common applications with only a single smart LED. With multiple smart LEDs, LiShield's coverage can be scaled up just like normal lighting (Section 4.12).

4.12 Conclusion and Social Impact

Considerations for High Peak Intensity. Considering the hardware capability and energy costs, we estimate the optimal LED peak intensity to be 20 kLux, and average intensity is 10 kLux with 0.5 duty cycle, which is an order of magnitude lower than outdoor intensity in a sunny day [165], and generally considered safe even for long activities [161]. Our smart LED is brighter than typical indoor lighting, which is usually less than 1 kLux. But we found the intensity is always acceptable by perception in our experiments, likely because the brightness perceived by human eyes and actual light intensity follow a logarithmic relationship. Since privacy protection has a higher priority than power savings, we expect a slight increase of illumination brightness is acceptable in the target use cases.

Multiple LEDs and Cameras. When a large indoor environment needs LiShield's protection, the smart LEDs can be deployed pervasively to cover the whole area, just like regular lighting. The availability of multiple LEDs can also increase the diversity of the protection since each of them can be controlled independently. We leave the optimization of such a multi-LED setup for future work.

With the presence of multiple unauthorized cameras, LiShield needs to ensure no additional information can be recovered by combining images across them, which may impose extra constraints on waveform design. Meanwhile, when multiple authorized cameras (sec:auth) are present, LiShield can serve them in a round-robin manner. Better strategies may require synchronization between cameras and we leave them for future work.

Attacker with Special Equipment. Global shutter cameras, ND filters (optical attenuators), and similar professional devices may compromise LiShield's protection. While this is inevitable, we note that such devices are usually bulky and costly, making them obtrusive and less accessible by everyday photographers. Thus, LiShield may still protect the privacy of its users by demotivating such attacks. An advanced version of LiShield that fully prevents such attacks will be left for our future work.

Recording a high-speed video (*e.g.*, 120 FPS) by advanced cameras will not significantly weaken LiShield's protection, as stripes across frames will be similar. High FPS also requires shorter exposure, which actually amplifies LiShield's effect. Along with the backup barcode protection, which is not affected by the camera's frame rate, the threat posed by a high-speed camera is limited.

Privacy protection in passive indoor environments has been an important but unsolved problem. In this chapter, we propose LiShield, which uses smart-LEDs and specialized intensity waveforms to disrupt unauthorized cameras, while allowing authorized users to record high-quality images and videos. We implemented and evaluated LiShield under various representative indoor scenarios, which demonstrates LiShield's effectiveness and robustness. We consider LiShield as the first exploration of automatic visual privacy enforcement and expect it can inspire more research along the same direction.

Chapter Review. In this chapter, we extended the concept of image effects to real photography. By building customized cameras and illumination sources, we can create and manipulate the structure of image appearances simultaneously to support real-world visual applications.

Acknowledgements. This chapter, in full, is a reprint of the material as it appears in International Conference on Mobile Computing and Networking 2017 and Communications of the ACM 2020 [210, 213]. Shilin Zhu, Chi Zhang, and Xinyu Zhang, Association for Computing Machinery, 2020. The dissertation author was the co-primary investigator and author of this paper. Chi Zhang was the other co-primary author who contributed equally to this work.

Disclaimer. The work presented in this chapter was equally contributed by both the author of this dissertation and his collaborator Chi Zhang at the University of Wisconsin, Madison. The author of this dissertation was primarily responsible for the algorithm part, and Chi Zhang was primarily responsible for the hardware and prototype part. The permission to include the joint work in this dissertation has been granted.

Chapter 5

Finale

After discussing a variety of image effects that are attainable with computer programs, we summarize the discoveries and scientific contributions in this last chapter. Although the future is normally unpredictable, my professional speculations are included in the end.

5.1 Conclusion and Open Problems

In this dissertation, we have systematically presented our research work on computing images under disparate illuminations. Our contributions can be encapsulated as follows:

- We raised the quality of radiance reconstruction in particle-based image synthesis by an efficient density estimation scheme;
- We introduced a learning-based path sampling strategy to compute global illumination on images at a higher rate of convergence;
- We extended the domain of imaging by innovating a coded photography framework to produce unconventional image appearances.

Our proposed methods are capable of reaching unprecedented performance levels in many cases. Nevertheless, limitations are inevitable, and there are unfortunately still numerous open questions.

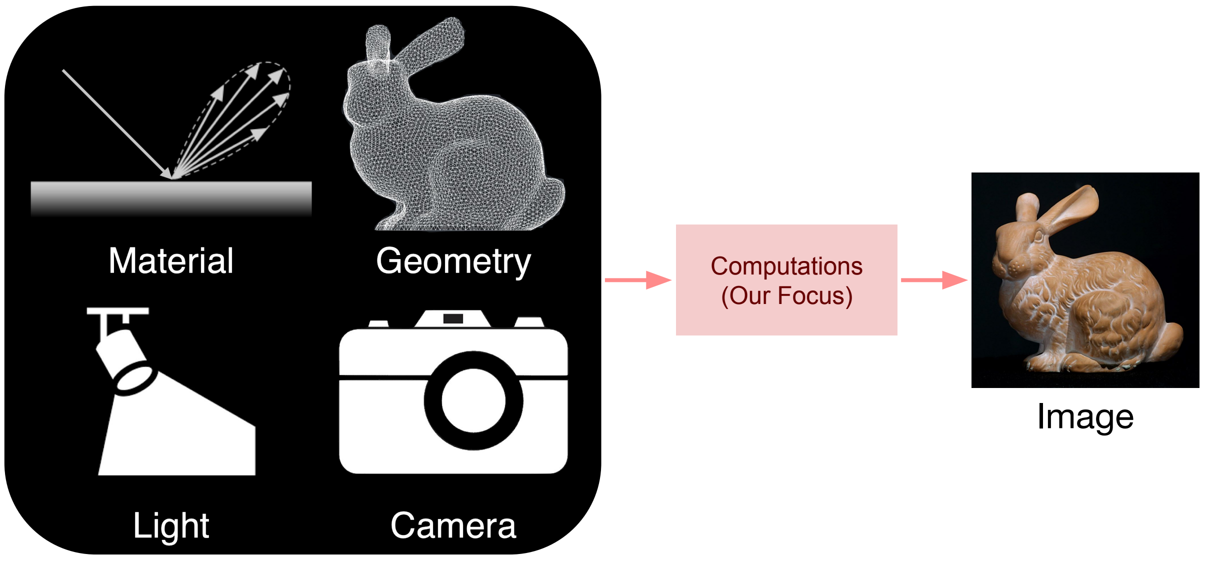


Figure 5.1. Summary of the image formation process. The scientific mission of our work is to innovate new computations to "paint" diverse illuminated effects on pictures that reflect our appearance expectation.

In the realm of realistic rendering, even the most state-of-the-art light transport solution has fear for the rapid growth of scene complexities. Taking the visual effect industry as a representative example, modeling and lighting artists have started to instance millions of objects and lights that regularly swallow hundreds of Gigabytes to a few Terabytes. Effective sampling and reconstruction save us some time, but the algorithmic essence is untouched. For a moment in the future, we may have to rethink the drawbacks of Monte-Carlo simulation and perhaps give birth to a brand-new method.

For the field of computational photography, the development of creative imaging techniques is majorly impeded by hardware limitations. Most working prototypes heavily rely on the use of professional cameras (e.g., DSLR) and bulky light sources (e.g., laser). Some particular designs even require extremely high-speed electronic circuits. Additionally, certain imaging steps are still dominated by sluggish mechanics such as the lens. Therefore, we must pay more attention to the practicality of our systems in future explorations.

We have witnessed the flourish of machine learning and its wide adoption on images in vision, graphics, and photography. Despite the successes we have accomplished, the computationally expensive nature of neural networks is obstructing the applicability to many problems, especially those with no access to GPUs. For future research involving deep learning, we ought to put extra focus on cost reduction possibly by introducing more domain-specific knowledge.

The continuous bloom of image-related technology makes me confident that the aforementioned open questions will be answered in the foreseeable future because we researchers always have the courage to step into the unknown.

5.2 The Future of Images: To 4D and Beyond

Images are significant to study because such 2D concept is closest to the pathway of human perception. Still, people are attempting to unlock more dimensions. For instance, 3D holography is an imaginative technique that permits the view from any direction by reconstructing the wavefront of light. Another example is light field cameras which equip extra optical components to capture the entire 4D light distribution by the plenoptic function. In terms of the time dimension, there are stacks of work on video analysis and extrapolation where computers can understand sequences of images and predict events that might occur. These research efforts are pulling our current impression of images to higher dimensions.

The concept of image is also being extended to other domains of science. For example, in astronomy, people have recently applied computational photography principles on telescopes to capture the first black hole image in human history. In medical science, we have started to see new volume rendering technologies that can help doctors examine various diseases such as tumors. Lastly, in planetary science, people are developing layered scanning methods to uncover the evolution of Earth. The development of imaging in computer science is responsible for the success in these fields.

It is my professional opinion that machines can never reach the same level of aesthetics as humans, regardless of how hard we seek in the future. Painting is a creative process by people like Vincent van Gogh to create pictures of outstanding artistry. Computers have no way to understand that, and they should not. It is important to realize that the primary goal of our research efforts on images is to bring the technology to serve, not to replace.

Without a doubt, it is a gift from nature that we have biological cameras to capture our world whenever the sun rises. We are witnesses of a golden age when the history of imaging is being rewritten. Through countless small steps of work like ours, a giant leap of visual computing is going to come.

Bibliography

[1] Digital Image Steganography: Survey and Analysis of Current Methods. *Signal Processing*, 90(3), 2010.

[2] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. I-Pic: A Platform for Privacy-Compliant Image Capture. In *Proceedings of ACM Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016.

[3] Saeed Al-Mansoori and Alavi Kunhu. Robust watermarking technique based on dct to protect the ownership of dubaisat-1 images against attacks. *International Journal of Computer Science and Network Security (IJCSNS)*, 12(6):1, 2012.

[4] Stephen J. Anderson and David C. Burr. Spatial and temporal selectivity of the human motion detection system. *Vision Research*, 25(8):1147 – 1154, 1985.

[5] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics (TOG)*, 36(4):97, 2017.

[6] Steve Bako, Mark Meyer, Tony DeRose, and Pradeep Sen. Offline deep importance sampling for monte carlo path tracing. In *Computer Graphics Forum*, volume 38, pages 527–542. Wiley Online Library, 2019.

[7] Mauro Barni. *Document and Image compression*. CRC press, 2006.

[8] BBC. The Camera Phone Backlash, 2004. URL http://news.bbc.co.uk/2/hi/uk_news/magazine/3793501.stm.

[9] Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/.

[10] Cheng Bo, Guobin Shen, Jie Liu, Xiang-Yang Li, YongGuang Zhang, and Feng Zhao. Privacy.Tag: Privacy Concern Expressed and Respected. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2014.

[11] Jérôme Boulanger. jboulanger.gmic. URL https://github.com/jboulanger/jboulanger-gmic.

[12] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[13] Herschel Clement Burstyn. Cinema anti-piracy measures, 2009. US Patent 7,634,089.

[14] Herschel Clement Burstyn, George Herbert Needham Riddle, Leon Shapiro, and David Lloyd Staebler. Method and apparatus for film anti-piracy, 2008. US Patent 7,324,646.

[15] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):98, 2017.

[16] Tung-Shou Chen, Chin-Chen Chang, and Min-Shiang Hwang. A virtual image cryptosystem based upon vector quantization. *IEEE transactions on Image Processing*, 7(10): 1485–1488, 1998.

[17] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020.

[18] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[19] Kashyap Chitta, Jose M Alvarez, and Martial Hebert. Quadtree generating networks: Efficient hierarchical scene parsing with sparse convolutions. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020.

[20] Cinetics. Axis360 Pro. http://cinetics.com/axis360-pro/.

[21] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1186–1195. ACM, 2005.

[22] Ronald R Coifman and David L Donoho. *Translation-invariant de-noising*. Springer, 1995.

[23] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3d filtering. In *Electronic Imaging*. International

Society for Optics and Photonics, 2006.

[24] Christos Danakis, Mostafa Afgani, Gordon Povey, Ian Underwood, and Harald Haas. Using a cmos camera sensor for visible light communication. In *Proc. of IEEE Globecom Workshops*, 2012.

[25] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[26] Tamara Denning, Zakariya Dehlawi, and Tadayoshi Kohno. In Situ with Bystanders of Augmented Reality Glasses: Perspectives on Recording and Privacy-mediating Technologies. In *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2014.

[27] Stavros Diolatzis, Adrien Gruson, Wenzel Jakob, Derek Nowrouzezahrai, and George Drettakis. Practical product path guiding using linearly transformed cosines. In *Computer Graphics Forum*, volume 39, pages 23–33. Wiley Online Library, 2020.

[28] F. Dufaux and T. Ebrahimi. Scrambling for Privacy Protection in Video Surveillance Systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8), 2008.

[29] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (TOG)*, 24(3): 1115–1126, 2005.

[30] Philip Dutré, Eric P Lafortune, and Yves D Willems. Monte carlo light tracing with direct computation of pixel intensities. 1993.

[31] Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. Frequency analysis and sheared reconstruction for rendering motion blur. In *ACM Transactions on Graphics (TOG)*, volume 28, page 93. ACM, 2009.

[32] LG Electronics. URL http://www.lg.com/ca_en/cell-phones/lg-D820-White-nexus5.

[33] Engadget. Japan's noisy iPhone problem, 2016. URL https://www.engadget.com/2016/09/30/japans-noisy-iphone-problem/.

[34] Michael Epstein and Douglas A Stanton. Method and device for preventing piracy of video material from theater screens, 2003. US Patent 6,529,600.

[35] James A Fancher, David H Sitrick, and Gregory P Sitrick. Movie film security system utilizing infrared patterns, 2003. US Patent 6,559,883.

[36] Iain Fergusson. External filters by iain fergusson. URL https://github.com/dtschump/

gmic-community/blob/master/include/iain_fergusson.gmic.

[37] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. Efficient network flooding and time synchronization with glossy. In *Proc. of ACM/IEEE IPSN*, pages 73–84, 2011.

[38] Forbes. Adventures in Self-Surveillance, aka The Quantified Self, aka Extreme Navel-Gazing, Apr. 2011.

[39] Cheng Fu, Shilin Zhu, Huili Chen, Farinaz Koushanfar, Hao Su, and Jishen Zhao. Simbnn: A similarity-aware binarized neural network acceleration framework. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 319–319. IEEE, 2019.

[40] Cheng Fu, Shilin Zhu, Hao Su, Ching-En Lee, and Jishen Zhao. Towards fast and energy-efficient binarized neural network inference on fpga. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 306–306, 2019.

[41] Nobuhiro Fujimoto and Hikari Mochizuki. 477 Mbit/s visible light transmission based on OOK-NRZ modulation using a single commercially available visible LED and a practical LED driver with a pre-emphasis circuit. In *National Fiber Optic Engineers Conference*, pages JTh2A–73. Optical Society of America, 2013.

[42] Nobuhiro Fujimoto and Shohei Yamamoto. The fastest visible light transmissions of 662 Mb/s by a blue LED, 600 Mb/s by a red LED, and 520 Mb/s by a green LED based on simple OOK-NRZ modulation of a commercially available RGB-type white LED using pre-emphasis and post-equalizing techniques. In *Optical Communication (ECOC), 2014 European Conference on*, pages 1–3. IEEE, 2014.

[43] Zhongpai Gao, Guangtao Zhai, and Chunjia Hu. The invisible qr code. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1047–1050. ACM, 2015.

[44] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[45] Iliyan Georgiev, Jaroslav Křivánek, and Philipp Slusallek. Bidirectional light transport with vertex merging. In *SIGGRAPH Asia 2011 Sketches*, page 27. ACM, 2011.

[46] Iliyan Georgiev, Jaroslav Krivánek, Tomas Davidovic, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6): 192–1, 2012.

[47] Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. Sample-based monte carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[48] Golf News Net. Why is it illegal in South Korea to silence mobile phone camera sounds?, 2015. URL https://thegolfnewsnet.com/golfnewsnetteam/2015/10/07/illegal-south-korea-silence-mobile-phone-camera-sounds-13503/.

[49] Rafael L Gomes, Luiz F Bittencourt, Edmundo RM Madeira, Eduardo Cerqueira, and Mario Gerla. Qoe-aware dynamic virtual network resource adaptation for eaas environment. In *Communications (ICC), 2015 IEEE International Conference on*, pages 6836–6841. IEEE, 2015.

[50] Dean K Goodhill and Ty Safreno. Method and apparatus for inhibiting the piracy of motion pictures, 2011. US Patent 8,018,569.

[51] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[52] Google. android.hardware.camera2. URL https://developer.android.com/reference/android/hardware/camera2/package-summary.html.

[53] Ben Graham. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[54] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[55] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[56] Phil Green and Lindsay MacDonald. *Colour engineering: achieving device independent colour*, volume 30. John Wiley & Sons, 2011.

[57] Stacey L. Gulick. Preventing Unauthorized Audio and Video Recording at Your Practice, 2004. URL http://medicaleconomics.modernmedicine.com/medical-economics/content/tags/hipaa/preventing-unauthorized-audio-and-video-recording-your-practice?page=full.

[58] Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. Primary sample space path guiding. In *Eurographics Symposium on Rendering*, volume 2018, pages 73–82. The Eurographics Association, 2018.

[59] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, volume 28, page 141. ACM, 2009.

[60] Toshiya Hachisuka and Henrik Wann Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics (TOG)*, 30(5):114, 2011.

[61] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, volume 27, page 130. ACM, 2008.

[62] Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. A progressive error estimation framework for photon density estimation. In *ACM Transactions on Graphics (TOG)*, volume 29, page 144. ACM, 2010.

[63] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)*, 31(6):191, 2012.

[64] Mark Harman. Open Camera. URL http://opencamera.sourceforge.net/.

[65] Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. Product importance sampling for light transport path guiding. In *Computer Graphics Forum*, volume 35, pages 67–77. Wiley Online Library, 2016.

[66] David Honzátko. CUDA implementation of BM3D. URL https://github.com/DawyD/bm3d-gpu.

[67] Roberto Hoyle, Robert Templeman, Steven Armes, Denise Anthony, David Crandall, and Apu Kapadia. Privacy Behaviors of Lifeloggers Using Wearable Cameras. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2014.

[68] Chin-Wei Hsu, Kevin Liang, Hung-Yu Chen, Liang-Yu Wei, Chien-Hung Yeh, Yang Liu, and Chi-Wai Chow. Visible light encryption system using camera image sensor. In *OptoElectronics and Communications Conference (OECC) held jointly with 2016 International Conference on Photonics in Switching (PS), 2016 21st*, pages 1–3. IEEE, 2016.

[69] Chunjia Hu, Guangtao Zhai, and Zhongpai Gao. Visible light communication via temporal psycho-visual modulation. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 785–788. ACM, 2015.

[70] Xuezhen Huang, Xin Sun, Zhong Ren, Xin Tong, Baining Guo, and Kun Zhou. Irradiance regression for efficient final gathering in global illumination. *Frontiers of Computer Science*, 9(3):456–465, 2015.

[71] Yuchi Huo, Rui Wang, Ruzahng Zheng, Hualin Xu, Hujun Bao, and Sung-Eui Yoon. Adaptive incident radiance field sampling and reconstruction using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 39(1):1–17, 2020.

[72] Image Team of the GREYC laboratory. G'MIC - GREYC's Magic for Image Computing. URL http://gmic.eu/.

[73] David S Immel, Michael F Cohen, and Donald P Greenberg. A radiosity method for non-diffuse environments. *Acm Siggraph Computer Graphics*, 20(4):133–142, 1986.

[74] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[75] Wenzel Jakob, Christian Regg, and Wojciech Jarosz. Progressive expectation-maximization for hierarchical volumetric photon mapping. In *Computer Graphics Forum*, volume 30, pages 1287–1297. Wiley Online Library, 2011.

[76] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Eurographics Workshop on Rendering Techniques*, pages 326–335. Springer, 1995.

[77] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.

[78] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995. ISSN 0097-8493.

[79] Yi Jiang, Ke Zhou, and Sheng He. Human visual cortex responds to invisible chromatic flicker. *Nature Neuroscience*, 10(5):657–662, 2007.

[80] Jaeyeon Jung and Matthai Philipose. Courteous Glass. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2014.

[81] James T Kajiya. The rendering equation. In *ACM SIGGRAPH computer graphics*, volume 20, pages 143–150. ACM, 1986.

[82] Nima Khademi Kalantari and Ravi Ramamoorthi. Deep high dynamic range imaging of dynamic scenes. *ACM Trans. Graph.*, 36(4):144–1, 2017.

[83] Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph.*, 34(4):122–1, 2015.

[84] Chun-Meng Kang, Lu Wang, Yan-Ning Xu, Xiang-Xu Meng, and Yuan-Jie Song. Adaptive photon mapping based on gradient. *Journal of Computer Science and Technology*, 31(1): 217–224, 2016.

[85] Anton S Kaplanyan and Carsten Dachsbacher. Adaptive progressive photon mapping. *ACM Transactions on Graphics (TOG)*, 32(2):16, 2013.

[86] Masayuki Karakawa. Laser video projection system and method with anti-piracy feature, 2006. US Patent 7,103,074.

[87] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. Deep convolutional reconstruction for gradient-domain rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[88] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[89] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics (TOG)*, 30(3):25, 2011.

[90] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (TOG)*, 33(4):1–13, 2014.

[91] Pradeep Kumar Jayaraman, Jianhan Mei, Jianfei Cai, and Jianmin Zheng. Quadtree convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 546–561, 2018.

[92] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. Luxapose: Indoor Positioning with Mobile Phones and Visible Light. In *Proc. of ACM MobiCom*, 2014.

[93] Eric P Lafortune and Yves Willems. Bi-directional path tracing. In *Compugraphics' 93*, pages 145–153, 1993.

[94] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015.

[95] Hui-Yu Lee, Hao-Min Lin, Yu-Lin Wei, Hsin-I Wu, Hsin-Mu Tsai, and Kate Ching-Ju Lin. Rollinglight: Enabling line-of-sight light-to-camera communications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 167–180. ACM, 2015.

[96] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[97] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):

1–16, 2019.

[98] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T. Campbell, and Xia Zhou. Real-Time Screen-Camera Communication Behind Any Scene. In *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2015.

[99] Zhenjiang Li, Cheng Li, Wenwei Chen, Jingyao Dai, Mo Li, Xiang-Yang Li, and Yunhao Liu. Clock Calibration Using Fluorescent Lighting. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, 2012.

[100] C. K. Liang, L. W. Chang, and H. H. Chen. Analysis and Compensation of Rolling Shutter Effect. *IEEE Transactions on Image Processing*, 17(8), 2008.

[101] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[102] M Ronnier Luo, Guihua Cui, and B Rigg. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, 26(5):340–350, 2001.

[103] Maxim Integrated Products, Inc. Why Drive White LEDs with Constant Current? Aug. 2004.

[104] Zhen Meng, Song Fu, Jie Yan, Hongyuan Liang, Anfu Zhou, Shilin Zhu, Huadong Ma, Jianhua Liu, and Ning Yang. Gait recognition for co-existing multiple people using millimeter wave sensing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 849–856, 2020.

[105] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J Mitra, and Leonidas J Guibas. Structurenet: hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG)*, 38(6):242, 2019.

[106] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[107] Blend Swap models, 2020. URL https://www.blendswap.com.

[108] Evermotion 3D models, 2020. URL https://evermotion.org.

[109] Turbo Squid 3D models, 2020. URL https://www.turbosquid.com.

[110] Thomas Müller. “practical path guiding” in production. In *ACM SIGGRAPH Courses:*

Path Guiding in Production, Chapter 10, pages 18:35–18:48, New York, NY, USA, 2019. ACM. doi: 10.1145/3305366.3328091.

[111] Thomas Müller, Markus Gross, and Jan Novák. Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum*, volume 36, pages 91–100. Wiley Online Library, 2017.

[112] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019.

[113] Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. Neural control variates. *ACM Transactions on Graphics (TOG)*, 39(6):1–19, 2020.

[114] Moni Naor and Adi Shamir. Visual Cryptography. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 1995.

[115] Narrative. Narrative Clip 2 Wearable HD Video Camera, 2016. URL http://getnarrative.com/.

[116] T. Naseer, J. Sturm, and D. Cremers. FollowMe: Person Following and Gesture Recognition With a Quadrocopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[117] Alaeddin Nassani, Huidong Bai, Gun Lee, and Mark Billinghurst. Tag It!: AR Annotation Using Wearable Sensors. In *SIGGRAPH Asia Mobile Graphics and Interactive Applications*, 2015.

[118] Elaine M. Newton, Latanya Sweeney, and Bradley Malin. Preserving Privacy by De-Identifying Face Images. *IEEE Transactions on Knowledge and Data Engineering*, 17(2), 2005.

[119] S. John Obringer and Kent Coffey. Cell Phones in American High Schools: A National Survey. *Journal of Technology Studies*, 33(1), 2007.

[120] ON Semiconductor. NCL30160: LED Driver, Constant Current Buck Regulator, 1.0 A. URL http://www.onsemi.com/PowerSolutions/product.do?id=NCL30160.

[121] Ryan S Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive wavelet rendering. *ACM Trans. Graph.*, 28(5):140, 2009.

[122] Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. Optix: a general purpose ray tracing engine. *Acm transactions on graphics (tog)*, 29(4):1–13, 2010.

[123] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Rendering Techniques 2000*, pages 11–22. Springer, 2000.

[124] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information Hiding–a Survey. *Proceedings of the IEEE*, 87(7), 1999.

[125] Philips Lighting B.V. Philips Hue. URL http://meethue.com.

[126] John D Price. Methods and apparatus for detection of motion picture piracy for piracy prevention, 2009. US Patent App. 12/322,915.

[127] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[128] Kun Qian, Shilin Zhu, Xinyu Zhang, and Li Erran Li. Robust multimodal vehicle detection in foggy weather using complementary lidar and radar signals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 444–453, 2021.

[129] QImaging. Rolling Shutter vs. Global Shutter, 2014. URL https://www.qimaging.com/ccdorscmos/pdfs/RollingvsGlobalShutter.pdf.

[130] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward Privacy-preserving Photo Sharing. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2013.

[131] Swati Rallapalli, Aishwarya Ganesan, Krishna Chintalapudi, Venkat N. Padmanabhan, and Lili Qiu. Enabling Physical Analytics in Retail Stores Using Smart Glasses. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2014.

[132] Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. Variance-aware path guiding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020)*, 39(4):151:1–151:12, July 2020. doi: 10.1145/3386569.3392441.

[133] Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Lanodn P. Cox. What You Mark is What Apps See. In *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016.

[134] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[135] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[136] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[137] Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. Robust denoising using feature and color information. In *Computer Graphics Forum*, volume 32, pages 121–130. Wiley Online Library, 2013.

[138] Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG)*, 2020.

[139] Mehul P Sampat, Zhou Wang, Shalini Gupta, Alan Conrad Bovik, and Mia K Markey. Complex wavelet structural similarity: A new image similarity index. *IEEE transactions on image processing*, 18(11):2385–2401, 2009.

[140] Yosef Sanhedrai, Ariel Schwarz, Liad Ben Yishai, and Zeev Zalevsky. System and method for preventing photography, 2007. US Patent App. 12/308,525.

[141] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. Photon differentials. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 179–186, 2007.

[142] Lars Schjøth, Jon Sporring, and O Fogh Olsen. Diffusion based photon mapping. In *Computer Graphics Forum*, volume 27, pages 2114–2127. Wiley Online Library, 2008.

[143] Oren Shani. Precise Time Synchronization Over WLAN. URL http://www.ti.com/lit/an/swaa162a/swaa162a.pdf.

[144] Ling Shao, Ruomei Yan, Xuelong Li, and Yan Liu. From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithms. *IEEE Transactions on Cybernetics*, 44(7):1001–1013, 2014.

[145] Peter Shirley, Bretton Wade, Philip M Hubbard, David Zareski, Bruce Walter, and Donald P Greenberg. Global illumination via density-estimation. In *Rendering Techniques' 95*, pages 219–230. Springer, 1995.

[146] David H Sitrick and James A Fancher. Anti-piracy protection system and methodology, 2004. US Patent 6,771,349.

[147] David H Sitrick and James A Fancher. Targeted anti-piracy system and methodology,

2007. US Patent 7,170,577.

[148] David H Sitrick and James A Fancher. System and methodology for validating compliance of anti-piracy security and reporting thereupon, 2011. US Patent 8,006,311.

[149] Fikret Sivrikaya and Bülent Yener. Time synchronization in sensor networks: a survey. *IEEE network*, 18(4):45–50, 2004.

[150] Vincent So. Anti-piracy image display methods and systems, 2009. US Patent 7,634,134.

[151] Social Pilot. 125 Amazing Social Media Statistics You Should Know, 2016. URL https://socialpilot.co/blog/125-amazing-social-media-statistics-know-2016/.

[152] Ben Spencer and Mark W Jones. Into the blue: Better caustics through photon relaxation. In *Computer Graphics Forum*, volume 28, pages 319–328. Wiley Online Library, 2009.

[153] Ben Spencer and Mark W Jones. Photon parameterisation for robust relaxation constraints. In *Computer Graphics Forum*, volume 32, pages 83–92. Wiley Online Library, 2013.

[154] Ben Spencer and Mark W Jones. Progressive photon relaxation. *ACM Transactions on Graphics (TOG)*, 32(1):1–11, 2013.

[155] S.S. Stevens. *Psychophysics: introduction to its perceptual, neural, and social prospects*. Transaction Publishers, 1975.

[156] STMicroelectronics. STM32F103C8. URL http://www.st.com/en/microcontrollers/stm32f103c8.html.

[157] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[158] Emil Tchoukaleysky. Digital cinema anti-piracy method and apparatus for liquid crystal projection systems, 2013. US Patent 8,559,791.

[159] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia. PlaceRaider: Virtual Theft in Physical Spaces With Smartphones. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

[160] Robert Templeman, Mohammed Korayem, David J. Crandall, and Apu Kapadia. PlaceAvoider: Steering First-Person Cameras away from Sensitive Spaces. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[161] Michael Terman and Jiuan Su Terman. Light therapy for seasonal and nonseasonal

depression: efficacy, protocol, safety, and side effects. *CNS spectrums*, 10(08):647–663, 2005.

[162] The GIMP Team. GIMP - GNU Image Manipulation Program. URL https://www.gimp.org/.

[163] Victor Tiscareno, Kevin Jonhson, and Cindy Lawrence. Systems and Methods for Receiving Infrared Data with a Camera Designed to Detect Images, 2011.

[164] CG Trader. 2020. URL http://www.cgtrader.com.

[165] PR Tregenza. The daylight factor and actual illuminance ratios. *Lighting Research & Technology*, 12(2):64–68, 1980.

[166] Khai N. Truong, Shwetak N. Patel, Jay W. Summet, and Gregory D. Abowd. *Preventing Camera Recording by Designing a Capture-Resistant Environment*. 2005.

[167] Christopher W. Tyler. Analysis of visual modulation sensitivity. II. Peripheral retina and the role of photoreceptor dimensions. *Journal of the Optical Society of America A*, 2(3): 393–398, 1985.

[168] H. van der Broeck, G. Sauerlander, and M. Wendt. Power driver topologies and control schemes for leds. In *Annual IEEE Applied Power Electronics Conference and Exposition*, 2007.

[169] Eric Veach. *Robust Monte Carlo methods for light transport simulation*, volume 1610. Stanford University PhD thesis, 1997.

[170] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995.

[171] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428. ACM, 1995.

[172] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37(4):124, 2018.

[173] Jirı Vorba. Bidirectional photon mapping. In *Proc. of the Central European Seminar on Computer Graphics (CESCG'11)*, 2011.

[174] Jiří Vorba and Jaroslav Křivánek. Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[175] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[176] Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. Path guiding in production. In *ACM SIGGRAPH Courses*, pages 18:1–18:77, New York, NY, USA, 2019. ACM. doi: 10.1145/3305366.3328091.

[177] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics (TOG)*, 16(3):217–259, 1997.

[178] Matt P Wand and M Chris Jones. *Kernel smoothing*. Chapman and Hall/CRC, 1994.

[179] Anran Wang, Chunyi Peng, Ouyang Zhang, Guobin Shen, and Bing Zeng. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, page 23. ACM, 2014.

[180] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing Zeng. Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 181–195. ACM, 2015.

[181] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions on Graphics (SIGGRAPH)*, 36(4), 2017.

[182] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6), 2018.

[183] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based CNNs with output-guided skip connections for 3D shape and scene completion. 2020.

[184] Zhou Wang and Eero P Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 2, pages ii–573. IEEE, 2005.

[185] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004.

[186] Allan G Weber. The USC-SIPI image database version 5. *USC-SIPI Report*, 315:1–24,

1997.

[187] Donald Henry Willis. Method, apparatus and system for anti-piracy protection in digital cinema, 2008. US Patent App. 12/736,774.

[188] H. James Wilson. You, By the Numbers. *Harvard Business Review*, Sep. 2012.

[189] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, and W. Burgard. Accurate Indoor Localization for RGB-D Smartphones and Tablets Given 2D Floor Plans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[190] Grace Woo, Andy Lippman, and Ramesh Raskar. VRCodes: Unobtrusive and Active Visual Codes for Interaction by Exploiting Rolling Shutter. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

[191] Lifan Wu, Ling-Qi Yan, Alexandr Kuznetsov, and Ravi Ramamoorthi. Multiple axis-aligned filters for rendering of combined distribution effects. In *Computer Graphics Forum*, volume 36, pages 155–166. Wiley Online Library, 2017.

[192] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[193] Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, and Rui Tang. Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation. *ACM Transactions on Graphics (TOG)*, 38(6):224, 2019.

[194] Nan Xu, Fan Zhang, Yisha Luo, Weijia Jia, Dong Xuan, and Jin Teng. Stealthy Video Capturer: a New Video-Based Spyware in 3G smartphones. In *Proceedings of the ACM conference on Wireless Network Security (WiSec)*, 2009.

[195] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4): 126, 2018.

[196] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[197] Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. Fast 4d sheared filtering for interactive rendering of distribution effects. *ACM Transactions on Graphics (TOG)*, 35(1):7, 2015.

[198] Zhe Yang, Yuting Bao, Chuhao Luo, Xingya Zhao, Siyu Zhu, Chunyi Peng, Yunxin Liu, and Xinbing Wang. ARTcode: Preserve Art and Code in Any Image. In *ACM International*

Joint Conference on Pervasive and Ubiquitous Computing (UbiComp), 2016.

[199] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7508–7517, 2020.

[200] Hong Heather Yu and Prabir Bhattacharya. Methods and apparatus for digital content protection, 2006. US Patent 7,006,630.

[201] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4471–4480, 2019.

[202] Guangtao Zhai and Xiaolin Wu. Defeating camcorder piracy by temporal psychovisual modulation. *Journal of Display Technology*, 10(9):754–757, 2014.

[203] Chi Zhang and Xinyu Zhang. LiTell: Robust Indoor Localization Using Unmodified Light Fixtures. In *Proc. of ACM MobiCom*, 2016.

[204] Chi Zhang and Xinyu Zhang. Pulsar: Towards Ubiquitous Visible Light Localization. In *Proc. of ACM MobiCom*, 2017.

[205] Lan Zhang, Cheng Bo, Jiahui Hou, Xiang-Yang Li, Yu Wang, Kebin Liu, and Yunhao Liu. Kaleido: You Can Watch It But Cannot Record It. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[206] Lan Zhang, Kebin Liu, Xiang-Yang Li, Cihang Liu, Xuan Ding, and Yunhao Liu. Privacy-friendly Photo Capturing and Sharing System. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2016.

[207] Quan Zheng and Matthias Zwicker. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, volume 38, pages 169–179. Wiley Online Library, 2019.

[208] Anfu Zhou, Guangyuan Su, Shilin Zhu, and HuaDong Ma. Invisible qr code hijacking using smart led. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–23, 2019.

[209] Shilin Zhu and Xinyu Zhang. Enabling High-Precision Visible Light Localization in Today's Buildings. In *Proc. of ACM MobiSys*, 2017.

[210] Shilin Zhu, Chi Zhang, and Xinyu Zhang. Automating visual privacy protection using a smart led. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 329–342, 2017.

[211] Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4923–4932, 2019.

[212] Shilin Zhu, Zexiang Xu, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. Deep kernel density estimation for photon mapping. In *Computer Graphics Forum*, volume 39. Wiley-Blackwell, 2020.

[213] Shilin Zhu, Chi Zhang, and Xinyu Zhang. Automating visual privacy protection using a smart led. *Communications of the ACM*, 63(2):81–89, 2020.

[214] Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. Photon-driven neural reconstruction for path guiding. *ACM Transactions on Graphics (TOG)*, 41(1):1–15, 2021.

[215] Shilin Zhu, Zexiang Xu, Tiancheng Sun, Alexandr Kuznetsov, Mark Meyer, Henrik Wann Jensen, Hao Su, and Ravi Ramamoorthi. Hierarchical neural reconstruction for path guiding using hybrid path and photon samples. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021.

[216] Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. In *Computer Graphics Forum*, volume 34, pages 667–681. Wiley Online Library, 2015.