UNIVERSIDADE FEDERAL FLUMINENSE

BRUNO AUGUSTO DORTA MARQUES

Lighting Estimation on Mixed Reality Using Deep Learning

NITERÓI 2020

UNIVERSIDADE FEDERAL FLUMINENSE

BRUNO AUGUSTO DORTA MARQUES

Lighting Estimation on Mixed Reality Using Deep Learning

Thesis presented to the Computing Graduate program of the Universidade Federal Fluminense in fulfillment of the requirements for the Ph.D. degree. Topic area: Visual Computing.

Orientador: ESTEBAN WALTER GONZALEZ CLUA

Co-orientadora: CRISTINA NADER VASCONCELOS

> NITERÓI 2020

Ficha catalográfica automática - SDC/BEE Gerada com informações fornecidas pelo autor

M3571 Marques, Bruno Augusto Dorta Lighting estimation on mixed reality using deep learning / Bruno Augusto Dorta Marques ; Esteban Walter Gonzalez Clua, orientador ; Cristina Nader Vasconcelos, coorientadora. Niterói, 2020. 117 f. : il. Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2020. DOI: http://dx.doi.org/10.22409/PGC.2020.d.39157923817 1. Realidade virtual. 2. Computação gráfica. 3. Inteligência artificial. 4. Aprendizado de máquina. 5. Produção intelectual. I. Gonzalez Clua, Esteban Walter, orientador. II. Nader Vasconcelos, Cristina, coorientadora. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

Bibliotecário responsável: Sandra Lopes Coelho - CRB7/3389

BRUNO AUGUSTO DORTA MARQUES

LIGHTING ESTIMATION ON MIXED REALITY USING DEEP LEARNING

Thesis presented to the Computing Graduate program of the Universidade Federal Fluminense in fulfillment of the requirements for the Ph.D. degree. Topic area: Visual Computing.

APPROVED BY

ESTEBAN WALTER GONZALEZ CLUA - Advisor, UFF the Shower CRISTINA NADER VASCONCELOS - Co-Advisor, UFF ALBERTO BARBOSA RAPOSO, PUC-Rio my unforming for ponnaturo ALINE MARINS PAES CARVALHO, UFF ANSELMO MONTENEGRO, UFF

LUIZ MARCOS GARCIA GONÇALVES, UFRN

Niterói 2020

To my family: Maria, João, and Rodrigo.

Acknowledgements

I want to thank my family: my father, João; my mother, Maria; and my brother, Rodrigo; for their unconditional love and support. It would be impossible to fulfill my personal and professional goals without them. I also thank every friend who stayed with me during the difficult moments in my life, in special, Tati, for the genuine friendship that accompanied me during a significant portion of my life.

I want to thank my advisor Esteban Clua, who always encouraged me to make great things and guided me to achieve a successful career. I also thank my co-advisor, Cristina Nader, and every research partner that I worked with during the development of my academic life. In special, I would like to thanks my co-workers Paulo Barchi, Rafael Drumond, Eder de Oliveira, João Coutinho, João Paulo Gois, Harlen Batagelo and many others who contributed to my work.

Finally, I also thank UFF for providing support and conditions to develop this thesis. Lastly, I thank CAPES and Nvidia for providing financial support for this work.

Resumo

Os recentes avanços em plataformas de realidade mista impulsionam o surgimento de novos paradigmas de interação. Em particular, Head-mounted Displays (HMDs) estão sendo desenvolvidos pela indústria para criar uma interface entre o mundo virtual e o mundo real, permitindo que os usuários interajam com elementos virtuais e reais presentes simultaneamente em uma única cena. Nessa composição de elementos reais e virtuais, discrepâncias perceptuais na iluminação de objetos podem ocorrer; essa diferença entre a iluminação do mundo virtual e a iluminação do mundo real é um problema referido como problema de incompatibilidade de iluminação. Neste trabalho propomos um framework de realidade mista para solucionar o problema de incompatibilidade de iluminação. O framework é baseado em estimar a iluminação de ambientes do mundo real e adaptar os elementos do mundo virtual de acordo com a iluminação estimada. Estimar a iluminação de uma cena real é uma tarefa difícil. Abordagens adotadas por trabalhos relacionados na literatura solucionam essa tarefa impondo fortes suposições sobre a cena, ou requerem o conhecimento prévio da cena, como por exemplo a geometria da cena, limitando a usabilidade dos métodos existentes para aplicações de realidade mista. Neste trabalho utilizamos uma abordagem baseada em deep learning para o desenvolvimento de três modelos para estimativa de iluminação. Os modelos são apropriados para o framework de realidade mista a não requerem conhecimento prévio da cena. O Point Light Source estimator e o Spherical Harmonics Light Probe estimator consideram que mãos humanas podem serem utilizadas como um *light-probe* da iluminação ambiente. Esta hipótese é explorada para inferir a iluminação ambiente utilizando fotografias em visualizações egocêntricas capturadas pela câmera localizada nos HMD's de realidade mista. O Environment Light Probe estimator faz uso de imagens de ambientes reais a fim de criar um modelo robusto de estimativa de iluminação ambiente capaz de estimar a iluminação ambiente utilizando uma fotografia em baixa faixa dinâmica do ambiente. Os métodos de estimativa de iluminação ambiente foram validados através de testes sintéticos e aplicações de re-iluminação de cenas em realidade mista. Trabalhos futuros são propostos afim de proporcionar melhorias que podem aumentar a imersão do usuário em aplicações de realidade mista.

Palavras-chave: Realidade Mista, Estimativa de Iluminação, Aprendizado Profundo, Redes Neurais Convolucionais

Abstract

Advances in mixed reality platforms are allowing new paradigms of interaction to emerge. In particular, Head-mounted Displays (HMDs) have been developed by the industry to create an interface between the virtual and the real world, allowing users to interact with both real and virtual elements located simultaneously in a single scene. In this composition of real and virtual elements, perceptual discrepancies in the illumination of objects may occur; this difference between the virtual world lighting and the real world environment lighting is a problem referred to as illumination mismatch problem. In this work, a mixed reality framework is presented to solve the illumination mismatch problem; the framework is based on estimating the real-world environment lighting and adapting the virtual elements to the estimated lighting. Estimating the environment lighting of a real-world scene is a difficult task, the approaches adopted by related works in the literature solves this task by either imposing strong assumptions about the environment or requiring prior knowledge of the scene, such as the scene geometry, thus limiting the usability of the existing methods for mixed reality applications. In this work, a deep learning approach is utilized to develop three lighting estimation models; the methods are suitable for the mixed reality framework and do not require prior knowledge of the scene. The Point Light Source estimator and the Spherical Harmonics Light Probe estimator considers that human hands can be used as a light-probe of the environment lighting. This hypothesis is explored to infer the environment lighting using pictures in egocentric views captured by cameras located in the mixed reality HMD. The Environment Light Probe estimator makes use of real environment light-probe images to create a robust lighting estimation model capable of estimate the environment lighting using a low-dynamic-range picture of the real environment. The lighting estimation methods were validated through synthetic tests and the relight of mixed reality scenes. Future works are proposed in order to provide enhancements that could increase the user immersion in mixed reality applications.

Keywords: Mixed Reality, Lighting Estimation, Deep Learning, Convolutional Neural Network

List of Figures

1.1	Approaches to solving the illumination mismatch	4
2.1	A virtual scene illuminated with different IBL environments [93]	9
2.2	High-dynamic-range (HDR) image vs low dynamic range (LDR) image	11
2.3	Light Probe image mappings.	12
2.4	Angular mapping relative directions	14
2.5	Scene illuminated by different spherical harmonics lighting environments .	15
2.6	Plot of the three bands spherical harmonic basis functions	19
2.7	Spherical harmonics light-probe image approximations	21
2.8	A virtual scene illuminated with different levels of spherical harmonics approximations.	22
2.9	Simple feedforward network representation	26
2.10	Hidden unity representation	27
2.11	Steps to perform a convolution operation in an image	31
2.12	Sparse and dense connectivity	33
2.13	Average pooling operation	34
2.14	The basic building block of a Residual Network	35
2.15	Residual Network Design : Stacking Residual building blocks	36
2.16	The basic building block of a Dense Convolutional Network	37
2.17	Dense Network Design : Stacking dense building blocks	38
2.18	The NASNet overall architecture	39
2.19	NASNet-A normal cell	40
2.20	NASNet-A reduction cell.	41

4.1	General mixed reality framework	56
4.2	Mixed Reality Framework overview	57
5.1	The Point Light Source Estimation Dataset scene setup.	65
5.2	Distribution of points on the surface of a sphere	67
5.3	The PLS Dataset sample images	68
5.4	PLS lighting estimation on synthetic input image	73
5.5	Lighting adjustment on Augmented Virtuality application	75
5.6	Point Light Source estimation with different lighting settings discretizations.	77
6.1	The SH Light Probe Dataset scene setup	82
6.2	The Dataset sample images	83
6.3	SHLP lighting estimation on synthetic input	87
6.4	SHLPE and PLSE lighting estimation on complex lighting scenarios	88
6.5	SHLPE and PLSE lighting estimation comparison	89
7.1	ELP estimator: architecture overview	92
7.2	Mixed-reality-view processing pipeline	95
7.3	Training and validation loss for the Environment Light Probe estimator.	97
7.4	Environment Light Probe estimator prediction results	99
7.5	Environment Light Probe estimator camparison with state-of-the-art	100
7.6	Environment Light Probe estimator on mixed reality scenes	101
7.7	Environment Light Probe estimator relighting of outdoor scenes	101
7.8	Environment Light Probe estimator limitations	102

List of Tables

1.1	Overview of the three lighting estimation methods developed	7
2.1	SH functions in cartesian coordinates	19
3.1	Lighting estimation methods comparison	51
5.1	Number of examples in the PLS dataset	67
5.2	The Point Light Source Estimator Network Architecture	70
5.3	PLS estimator: lighting estimation on the test dataset. \ldots \ldots \ldots	72
5.4	PLS estimator: lighting estimation on the synthetic test	74
6.1	The Spherical Harmonics Light Probe Estimator Network Architecture	85
6.2	SHLP estimator: lighting estimation on syntethic test	88
7.1	ELP estimator: lighting estimation on the test dataset	98

List of Abbreviations and Acronyms

ANN	:	Artificial Neural Network;
AR	:	Augmented Reality;
AV	:	Augmented Virtuality;
CNN	:	Convolutional Neural Network;
CPU	:	Central Processing Unit;
DDR	:	Double Data Rate;
ELP	:	Environment Light Probe;
GDDR	:	Graphics Double Data Rate;
GPS	:	Global Positioning System;
GPU	:	Graphics Processing Unit;
HDR	:	High Dynamic Range;
HMD	:	Head Mounted Displays;
IBL	:	Image Based Lighting;
LDR	:	Low Dynamic Range;
MR	:	Mixed Reality;
MSE	:	Mean Squared Error;
NAS	:	Neural Architecture Search;
PLS	:	Point Light Source;
ReLu	:	Rectified Linear unit;
ResNet	:	Residual Network;
RGB	:	Red, Green, Blue;
RMSE	:	Root Mean Squared Error;
SGD	:	Stochastic Gradient Descent;
SHLP	:	Spherical Harmonics Light Probe;
SH	:	Spherical Harmonics;
VR	:	Virtual Reality.
XR	:	eXtended Reality.

Contents

1	Intr	oductio	n	1
	1.1	Mixed	Reality	1
	1.2	Proble	em Statement: Illumination mismatch	3
	1.3	Contri	ibutions	6
	1.4	Thesis	Structure	8
2	The	oretical	Background	9
	2.1	Image	-based Lighting	9
		2.1.1	Light Probe Image	11
		2.1.2	Latitude-longitude Map	12
		2.1.3	Angular Map	13
	2.2	Spheri	cal Harmonics Lighting	15
		2.2.1	Signal Reconstruction	16
		2.2.2	Spherical Harmonics Functions	17
		2.2.3	Spherical Harmonics Light Probe	20
	2.3	Machi	ne Learning	23
		2.3.1	Learning Process	23
		2.3.2	Stochastic Gradient Descent	24
		2.3.3	Feedforward Neural Network	25
		2.3.4	Activation Functions	28
		2.3.5	Back-Propagation	29
		2.3.6	Feature Learning	29

	2.4	Convo	lutional Neural Network	30
		2.4.1	Convolution Layer	30
		2.4.2	Pooling Layer	33
		2.4.3	Residual Network Architecture	34
		2.4.4	Dense Convolutional Network Architecture	36
		2.4.5	Neural Architecture Search	38
3	Ligh	ting Es	timation Related Works	42
	3.1	Overv	iew	42
	3.2	Device	e-based light probe	43
	3.3	Know	geometry	45
	3.4	Doma	in-Specific Methods	47
	3.5	Surfac	e reflectance and lighting	48
	3.6	Relate	ed Works Summary	49
4	Ligh	ting Es	timation on Mixed Reality	52
	4.1	Light	Representation	53
	4.2	Huma	n Hands Light-Probe Hypothesis	55
	4.3	Mixed	Reality Consistent Lighting Framework	55
		4.3.1	Input Processing	57
		4.3.2	Lighting Estimation	59
		4.3.3	Lighting Adjustment	60
		4.3.4	Hands Montage	60
	4.4	Deep 1	Learning Approach to Lighting Estimation	61
5	Poin	nt-Light	Source Estimator	62
	5.1	Input	Data	63
	5.2	Point	Light Source Dataset	64

	5.3	Point	Light Source Estimator Architecture	69
	5.4	The R	esidual Network Training	70
	5.5	Exper	iments and Results	71
		5.5.1	Training Time and Accuracy	71
		5.5.2	Synthetic Scenario	72
		5.5.3	Augmented Virtuality Scenario	74
		5.5.4	Level of discretization	76
6	Sph	erical H	Iarmonics Light-Probe Estimator	79
	6.1	Input	Data	80
	6.2	Spheri	ical Harmonics Light Probe Dataset	81
	6.3	Spheri	ical Harmonics Light Probe Estimator architecture	84
	6.4	Model	Training	85
	6.5	Exper	iments and Results	86
		6.5.1	Training Time and Accuracy	86
		6.5.2	Synthetic Scenario	87
		6.5.3	Spherical Harmonics vs Point Light Estimators	89
7	Env	ironme	nt Light-Probe Estimator	90
		7.0.1	Environment Lighting Estimation	91
		7.0.2	Lighting Estimation CNN Architecture	92
	7.1	Learni	ing from HDR panoramas	94
		7.1.1	Laval Indoor HDR Database	95
		7.1.2	Mixed-reality-views	95
	7.2	Result	s and Experiments	96
		7.2.1	Performance	97
		7.2.2	Lighting Estimation	97

		7.2.3	Scenes Relight	101
	7.3	Discus	sion	102
		7.3.1	Limitations and Future Work	103
8	Fina	l Rema	rks	104
	8.1	Mixed	Reality Framework	104
	8.2	Point-	Light Source Estimator	105
	8.3	Spheri	cal Harmonics Light-Probe Estimator	105
	8.4	Enviro	onment Light-Probe Estimator Method	106
	8.5	Highli	ghts	107
	8.6	Future	e Works	108
Re	eferen	ces		109

Chapter 1

Introduction

1.1 Mixed Reality

Mixed reality (MR) [82] is the mixture of virtual reality environment and the real world. Several applications can benefit from mixed reality because of the increased user's immersion in the simulated environment. This increased immersion can be achieved due to improvements in the realism of the simulated environment and the creation of a personalized experience.

The term mixed reality has many distinct definitions [105]. In this work, we use the definition that is part of the Reality-Virtuality continuum by Milgran *et al.* [82]. The mixed reality spectrum defines a range of environments between an entirely real-world scene and a completely virtual environment.

In this spectrum, it is possible to describe the augmented reality (AR) environment, where most of the environment is composed of objects from the real world. In the AR environment, virtual objects are inserted in the real environment allowing user's interactions with both virtual and real-world objects.

At the other end of the mixed reality spectrum, there is the augmented virtuality (AV), where most of the environment is composed of virtual objects. Real objects are inserted in this virtual environment and can interact with the user and the virtual world. An additional form of augmented virtuality is the usage of real-world information, such as movement sensors, GPS location, and weather information, to increase realism in the virtual world.

In the extrem of the Reality-Virtuality continuum is the real environment consisting exclusively of real objects. In the other end of the continuum is the virtual environment consisting exclusively of virtual objects. Virtual environments are an essential part of the virtual reality (VR) applications.

AR and VR applications usually make use of the user's information, such as the user's location, movements, and image, to bring a personalized experience to the users.

The personalized experience has been explored by the entertainment industry to attract new users to video-games (Nintendo Wii, Microsoft Kinect) and improve the user's interaction in home entertainment systems. The advances in MR and VR technology have a huge potential to increase even more the personalization of the user's experience by introducing the user as an active character in the simulation.

Recent developments in Virtual Reality (VR) technologies are making low-cost headmounted displays (HMDs) available to the public. The HMDs manufacturers are investing significant technical and monetary resources in creating and distributing immersive content for home entertainment, including video games and cinema.

The increasing popularity of VR and MR is reflected by the worldwide revenues for the mixed reality and virtual reality market that is expected to grow from \$5.2 billion in 2016 to more than 162 billion in 2020 [21].

Video games and other simulation contents for VR and MR tends to treat the user as the main character of the retreated history. The usual representation of the user is a virtual character (an animated 3D model); this virtual identity is called the avatar.

For VR and MR applications, the avatar is almost exclusively seen from the firstperson point of view, meaning that a camera is positioned in the eyes of the avatar, and what is seen by the camera is the representation of what the avatar is observing in the environment. The first-person point of view also means that most of the time the upper limbs (hands and arms) are the only visible parts of the avatar.

A significant improvement in the user's immersion in MR applications can be achieved by substitution of the avatar's upper limbs to the user's real upper limbs. This substitution would imply that the user can see the exact real appearance and movement of his arms and hands, including skin color, geometry, and lighting in the mixed reality simulation. Hence, increasing the user's personalization of the experience dramatically.

The substitution of the avatar's synthetic upper limbs to the real user's upper limbs is not straightforward. The projection of a real-world 2D footage containing the user's body, captured from a color camera, into the mixed reality environment is a possible approach. The captured footage containing the user's upper limb is projected into the virtual camera plane; it is necessary to pre-process the captured footage to remove unwanted objects from the real scene environment.

This approach is capable of accurately represent the user's physical attributes in the virtual environment but fails to merge the real world and virtual world appearance due to the different lighting conditions. We call this difference in the lighting condition between the real and virtual environments as the illumination mismatch problem. So we propose in this thesis methods that estimate the lighting of the real-world environment in an effort to mitigate the illumination mismatch problem in mixed reality applications.

1.2 Problem Statement: Illumination mismatch

The illumination mismatch problem can affect the way that the user perceives the scene. Once the user distinguishes different lighting conditions in the real objects and the virtual scene, he/she loses the sensation of belonging to the scene, leading to a decrease in the user's immersion.

The illumination mismatch problem can be solved by adjusting the illumination of the real, virtual, or both environments. In this thesis work, we choose to concentrate on the task of estimating the lighting of the real-world environment and adjusts the lighting in the virtual world environment.

Figure 1.1 shows an example of two different approaches: the real-world relight approach, and the virtual world relights approach. The middle row of the image gives an example of a real-world environment picture, a render of the virtual objects, and a mixed reality environment, respectively. Considering that the real-world lighting and the virtual environment lighting does not match, the illumination mismatch is clearly visible in the mixed reality environment (middle row).

The real-world relight approach (top row of Figure 1.1) consist of modifying the picture of the real-world environment to mimic the virtual world lighting setting. The virtual world relight approach (bottom row of the Figure 1.1) consists of modifying the lighting settings of the virtual environment to mimic the real-world lighting, the virtual objects are rendered with the real-world environment lighting to produce a plausible mixed reality environment.

The illumination of the virtual world is known a priori in the virtual environments. The information of the light source's position and their properties are required to render



Figure 1.1: Approaches to solving the illumination mismatch: The illumination mismatch can be solved by either adapting the real-world environment lighting picture to match the virtual environment lighting (top row of the image) or by adjusting the virtual world environment lighting to match the real-world environment lighting (bottom row of the image).

the virtual environment correctly. Since every light source is well known, adjusting their properties is also straightforward.

The illumination in the real environment is unknown to the simulation. There is no available information about lighting conditions such as light source position, direction, intensity, or color in the real environment. Estimating these characteristics is not a straightforward task, and can be particularly critical in cases of dynamic changes in the lighting conditions. Changing the lighting configuration of a real-world picture with unknown illumination is also a difficult task. This task would require the reconstruction of the real-world environment to recreate the shading of all the objects affected by the changes in the environment lighting.

In a Mixed Reality scenario, all the visual information from the user's real-world environment comes from pictures (RGB images) taken by a camera. An RGB image does not provide explicit information about the geometry, or the material reflectance of the objects in the scene. Therefore, all the real-world environment lighting information needs to be extracted from pictures of the environment.

One important action to mitigate the illumination mismatch problem is to recognize (or estimate) the lighting condition of the real-world environment. Once the lighting condition in the real-world environment is known, it is plausible to use this information to match the virtual and real environment lighting by changing the lighting setup of the virtual environment.

The matching of lighting conditions in both real and virtual environments is beneficial to the mixed reality spectrum in both ends. On the augmented reality end, virtual objects inserted in the real scenario can have a realistic appearance and behave like a real object. On the augmented virtuality end, a real object can seamlessly be inserted in the virtual environment.

In order to have a practical mixed reality application, the lighting estimation cannot be onerous for the user; thus, it should not requires additional sensors, hardware, or complicated setup procedures. By the nature of such applications, which presupposes the user's movement in the environment during the interactive simulation, the lighting estimation process should be computationally fast enough to achieve an interactive rate and recognize changes in the illumination.

The lighting estimation is a pattern recognition task that can be treated by machine learning algorithms. Among machine learning approaches that could tackle this task, deep learning algorithms have been responsible for most of the successful methods to classify or recognize patterns in images and videos.

Artificial neural network (ANN) algorithms (referred to as neural networks) are specialized algorithms for pattern recognition in data. These algorithms are inspired by the physiological structure of the human brain, where an intricate network of connections between cells called neurons acts as a mechanism to learn the solution of a given task. In the computational neural network, data processing cells called artificial neurons are connected in a specific way, forming structures referred to as layers. Classically, neural network algorithms made use of architectures containing few layers of neurons.

Advances in processing power, in particular by the development of Graphics Processing Units (GPUs), and the high availability of data have driven the ANNs researchers to increase the number of layers in the ANNs architectures. The possibility of an increased number of layers, along with more complex mathematical operations, establishes a new paradigm of learning algorithms that are capable of learning efficient representations of the data. Deep learning is a concept that defines techniques that applies neural network architectures with multiple layers that are capable of learning a suitable representation of the data to solve machine learning tasks. Considering the nature of the lighting estimation problem as a pattern recognition task, we explore deep learning algorithms to solve the lighting estimation task.

1.3 Contributions

We developed three strategies to estimate the lighting in the real-world environment based on a single picture of the real-world environment. The enhanced contribution of the developed methods is a **novel strategy based on deep learning to estimate lighting from a single picture as input**, our methods have the following characteristics:

- Our methods are notably more practical than other existing approaches regarding mixed reality applications. Our developed proposal does not requires special devices such as depth cameras, fish eyes lenses or passive probes inserted in the scene and do not requires previous knowledge of the scene's geometry;
- Our methods are suitable for both indoor and outdoor environments.
- Our methods have fast inference, making it suitable for interactive environments.
- Our method utilizes lighting representations that posses easy integration with current popular game engines such as Unreal Engine and Unity.

The results presented in this thesis have been published at Computer and Graphics journal, SBGames 2018 and at GRAPP 2018. The three methods for lighting estimation developed in this thesis are summarized in the Table 1.1:

- The point light source estimator is a method that estimates point light sources in a discretely distributed space.
- The spherical harmonics light probe estimator estimates spherical harmonics lighting coefficients that represents the environment lighting.
- The environment light-probe estimator estimates the environment lighting from a single image of the scene that is not constrained by any physical light probe.

Table 1.1: Overview of the three lighting estimation methods developed.

Hands based, discrete lighting estimation					
Point light source estimator Deep light source estimation for mixed reality [77]					
Objective	Method	Conclusions			
To investigate the lighting estimation based on user's hands informations; the lighting is modelled by point light sources discretely distributed in the space.	A 50-layer residual neural network estimator solving a classification problem. Input : RGB image containing user's hands. Output: Light position in a discretized space.	Qualitative and quantitative tests show that it is possible to recover the scene's lighting from the user's hands. The technique is limited by complex lighting scenarios like multiple light sources and global illumination			
На	nds based, area lighting estima	tion			
Spheric Deep spherical harmonic	al harmonics light probe e s light probe estimator for	stimator • mixed reality games [76]			
To develop a robust lighting estimator capable of estimate complex lighting scenarios subjects to multiple light sources with non-uniform intensity across the scene.	A 50-layer densely connected convolutional network estimator solving a regression problem. Input : RGB image containing user's hands. Output: Spherical Harmonics lighting encoded in 9 coefficients.	The resulting estimator improves on previous method by providing a more accurate lighting estimation that works on complex lighting scenarios including multiple light sources. The resulting estimation does not account for lighting color.			
Sco	ene based, area lighting estima	tion			
Env	vironment light-probe estin	nator			
To develop a lighting estimator capable of recognize the environment lighting based on a single image of the scene. The estimator should not be constrained by physical light probes.	A custom deep neural network architecture solving a regression problem. The input is a low dynamic range, RGB image of the real scene and the output is a sh lighting encoded in 27 coefficients.	The custom neural network was able to estimate the environment lighting, including ambient light color and intensity. Relight of mixed reality scenes demonstrate the technique under real usage case.			

1.4 Thesis Structure

Chapter 2 presents some key background concepts on Machine Learning, Deep Learning, Convolutional Neural Network, and The Residual Network Framework. Furthermore, a brief review is presented regarding image-based lighting and the background theory on Spherical Harmonics (SH) as a basis function to represent the environment lighting.

In the chapter 3 we present a literature review of the current techniques for lighting estimation. The discussion is centered around techniques for environment lighting estimation based on passive and active probes, geometry-based techniques, and methods that are tailored for outdoor environments.

In Chapter 4 the mixed reality framework is presented; the framework offers a methodology to circumvent the illumination mismatch problem through the usage of lighting estimation models. This chapter also discusses the hand-based environment light probe hypothesis that is assumed in some environment lighting models developed in this thesis.

In Chapter 5 we present the Point Light Source estimator, which is an environment lighting estimation model that estimates a single light source position from a single 2D image. This method allows applications to adjust the lighting in the virtual scene by positioning (or moving) a single point light in the virtual world.

Chapter 6 presents the Spherical Harmonics Light Probe (SHLP) estimator that estimates the SH Coefficients that represents a light probe of the scene encoded in a Spherical Harmonics Basis. This method is capable of estimate multiple light sources in the scene with different intensities and configurations.

Chapter 7 presents the Environment Light Probe estimator that estimates the environment light using a single picture of the real-world environment. This method does not make use of human hands as light-probes of the environment lighting. Thus it is suitable for a more widespread range of applications.

Chapter 8 presents a list of the key contributions of this thesis, including the mixed reality framework that is the methodology utilized for solving the illumination mismatch problem. A discussion is made about the characteristics and advantages of the three developed lighting estimation models. To conclude this thesis, this final chapter addresses the current and future work in the lighting estimation problems, including possible advances that can improve the user's immersion and experience in Mixed Reality applications.

Chapter 2

Theoretical Background

In this chapter we discuss the background knowledge required for this thesis development. We first review computer graphics topics, starting from image-based lighting that includes lighting representation and some fundamentals of image-based rendering. Then we describe spherical harmonics lighting that is an essential subject of this thesis. Later on in this chapter, we discuss the machine learning topics starting from artificial neural networks, extending to convolutional neural networks.

2.1 Image-based Lighting



(a) Sunset on a beach environ- (b) Inside a cathedral environ- (c) outside in a cloudy day enment. wironment.

Images can be used as a source of illumination for computer graphics. The usage of an image as the primary lighting environment and the accompanying rendering techniques are called image-based lighting (IBL). This technique is extensively used to create realistic motion-picture visual effects that combine computer-generated objects with real filmed footage. In a movie set, an environment-map can be easily acquired through multiple pictures of the physical environment. Figure 2.1 shows an example of a virtual scene illuminated by different image-based lighting environments.

Figure 2.1: A virtual scene illuminated with different IBL environments [93].

The most apparent benefit of using IBL for computer graphics is the increased level of realism of the generated scene. The accurate and consistent representation of the shadows, reflections, and shading complexities produces realistic images in which objects appear to belong to the scene that is lighting them.

The IBL is an element of the physically-based rendering (PBR). PBR is a collection of techniques, based on the principles of physics, to represent the interaction of the lighting in the physical world. The reflectance equation is a fundamental component in the PBR. For a given surface, the reflectance equation expresses the sum of the reflected radiance of a point p in outgoing direction w_o . The reflectance equation is defined as follows:

$$L_{0}(p, w_{0}) = \int_{\omega} f_{r}(p, w_{i}, w_{o}) L_{i}(p, w_{i}) n \cdot w_{i} dw_{i}$$
(2.1)

where L_i is the incoming radiance coming from the incident direction w_i in the point pof the surface; the term $n \cdot w_i$ modulates the outward radiance at the point p based on the angle between the incident direction w_i and the surface orientation defined by the surface normal n; and f_r is the bidirectional reflective distribution function (BRDF) [58] that represents the interaction between the incoming radiance and the surface's material properties.

The environment lighting in the PBR is represented by the L_i term in the reflectance equation. The environment map stores discretized samples of the scene radiance for a large number of directions w_i , centered in a given position p. The environment map is defined by a transformation that maps the radiance samples in a given angular direction (ϕ, θ) to texture coordinates (u, v).

A appropriate real-world image can be used as environment map if it contains information about the color and the full range of intensities for all the light sources. A low-dynamic-range (LDR) image, such as the images captured by consumer cameras and smartphones are not capable of representing the full dynamic-range of light. The image captured by those devices employs a pixel with a byte representing each one of the red, green, and blue channels, leading to only 256 possible values per color channel for each pixel in the image. This small number of variations for the colors make LDR images unsuitable for many scenes, making a poor representation of the environment radiance. Figure 2.2 illustrates the lack of information in an LDR image when utilized as a light source for image-based rendering, where the lack of information led to low fidelity at high-illuminated areas as well as the low resolution in shadowed areas. HDR images are essential for the IBL process, as a consequence, both the images and the process are



Figure 2.2: Sphere render with high-dynamic-range (HDR) image vs low dynamic range (LDR) image [90]

commonly referred to as HDRI, for high dynamic range imagery. For more information about HDR, Reinhard et al. [93] provide extensive details about HDR images capturing, processing, and other useful applications.

Another key aspect of the images utilized in IBL is the omnidirectional view. That is, the image should contain information about all the directions from a particular point in space. The omnidirectional nature is essential for IBL because the light coming from every direction contributes to the appearance of real-world objects. The radiance of the environment is a sum of contributions along all the incoming direction w_i in the L_i term of the Equation 2.1. Those omnidirectional HDR images captured of real-world environments are commonly referred to as light probe images.

2.1.1 Light Probe Image

Capturing the light probe image is a stage of the IBL process that should be carefully planned to produce high-quality images. Several techniques exist, each one with their advantages and disadvantages with respect to capturing and processing steps. There are techniques based on picture-taking and processing methods such as photographing mirror ball and tiled photographs; there are also techniques based on fish-eye lenses and panoramic cameras. More information about those light probe capturing techniques and their details can be read in the literature (Reinhard et al.).



(a) Latitude-longitude map.

(b) Angular map.

Figure 2.3: Light Probe image mappings.

2.1.2 Latitude-longitude Map

An omnidirectional image mapping is utilized to store the light probe image. Among several existing mapping available approaches, we used in this dissertation the latitudelongitude and the angular map. Image 2.3 is an example of a light probe in both angular and latitude-longitude image mappings.

Those mappings are defined by functions that map the image (u, v) coordinates to the corresponding unit direction in the world $D = (D_x, D_y, D_z)$ coordinates and the inverse operation (mapping world coordinates to image coordinates). Throught this thesis dissertation, a right-handed coordinate system is assumed.

The omnidirectional image can be seen as a spherical representation of the surrounding world, where the objects and lights of the world are distant relative to the diameter of the sphere.

The latitude-longitude mapping essentially flattens the sphere into a rectangular area. The rectangular image domain is defined as: $u \in [0, 2], v \in [0, 1]$. The mapping equation for world-to-image is:

$$(u, v) = (1 + \frac{1}{\pi} \arctan_2(D_x, -D_z), \frac{1}{\pi} \arccos D_y).$$
(2.2)

The inverse operation, image to world mapping equations are:

$$(\theta, \phi) = (\pi(u-1), \pi v),$$
 (2.3)

$$(D_x, D_y, D_z) = (\sin\phi\sin\theta, \cos\phi, -\sin\phi\cos\theta).$$
(2.4)

The $\arctan_2(\beta, \alpha)$ is defined as follows:

$$\operatorname{arc} \tan_{2}(\beta, \alpha) = \begin{cases} \operatorname{arc} \tan(\frac{\beta}{\alpha}) & \text{if } \alpha > 0 \\ \operatorname{arc} \tan(\frac{\beta}{\alpha}) + \pi & \text{if } \alpha < 0 \text{ and } \beta \ge 0 \\ \operatorname{arc} \tan(\frac{\beta}{\alpha}) - \pi & \text{if } \alpha < 0 \text{ and } \beta < 0 \\ \frac{\pi}{2} & \text{if } \alpha = 0 \text{ and } \beta > 0 \\ -\frac{\pi}{2} & \text{if } \alpha = 0 \text{ and } \beta < 0 \\ 0 & \text{if } \alpha = 0 \text{ and } \beta = 0. \end{cases}$$
(2.5)

The latitude-longitude mapping implies that the azimuthal direction is mapped to the horizontal coordinate, while the directional elevation is mapped to the vertical coordinate of the image. Hence, this mapping leads to a convenient representation of the light probe image providing the least distortion in regions near the horizon. Furthermore, the lighting map can be rotated around the y-axis by simply translating the image horizontally. The latitude-longitude image has a 2:1 aspect ratio corresponding to 360° by 180° , and it fully utilizes the rectangular area of the image. The major disadvantage of this mapping is that it is not equal-area; the regions starting from the horizon towards the top and vertical directions are progressively more oversampled. This overrepresentation is inversely proportional to the cosine of the latitude phi. Therefore to find the average pixel value of this light probe, it is necessary to multiply the image values by the vertical cosine falloff function $\cos(\phi)$ and then compute the average value.

The latitude-longitude format, along with cube maps, is frequently utilized in game engines and 3D animation tools. Figure 2.3a illustrates an example of a light probe in the latitude-longitude mapping format.

2.1.3 Angular Map

The angular map utilizes a circular portion of the square image. Thus the image domain is $u \in [0, 1], v \in [0, 1]$. Due to the use of a circular portion of the square image, this mapping is often confused with the ideal-mirrored-sphere mapping, although the angular map samples the regions around the edge of the image differently, avoiding undersampling. This map preserves the distance from the center point to the edges of the image; this distance is proportional to the angle between the straight-forward direction and the direction in world coordinates. In the angular mapping, the straight-forward looking direction is located in the center of the image, while the straight-left and straight-up looking direction is located halfway to the edges of the image. Figure 2.4 illustrates these locations in the angular map.

The mapping equations in the angular map, for world-to-image, are:

$$r = \frac{\arccos(-D_z)}{2\pi\sqrt{D_x^2 + D_y^2}} \tag{2.6}$$

$$(u,v) = \left(\frac{1}{2} - rD_y, \frac{1}{2} + rD_x\right) \tag{2.7}$$

The inverse operation, image-to-world mapping equations are:

$$(\theta, \phi) = \left(\arctan_2(-2v+1, 2u-1), \pi\sqrt{(2u-1)^2 + (2v-1)^2} \right)$$
(2.8)

$$(D_x, D_y, D_z) = (\sin\phi\cos\theta, \sin\phi\sin\theta, -\cos\theta)$$
(2.9)

Figure 2.3b illustrates and example of a light probe in the angular mapping format. One practical disadvantage of the angular map for computer graphics applications is that not the entire domain of the image contains valid information; thus, a substantial portion of GPU memory is wasted when allocating a square texture to accommodate the angular map image.



Figure 2.4: Angular mapping relative directions. The center point (in red) is the straight-forward looking direction, the left point (in blue) is the straight-left looking direction, the top point (in green) is the straight-up looking direction in the real world coordinates.

2.2 Spherical Harmonics Lighting

In the rendering equation, the calculation of irradiance at a particular point on the surface takes into account the contribution of the incoming radiance over all directions. This incoming radiance determines the lighting incident at this particular point of the surface.

Due to the directional nature of the incident lighting term in the rendering equation, it is possible to model the incoming radiance in a given point as functions on the surface of a unit sphere, centered at the point's position.



Figure 2.5: Scene of the movie Avatar illuminated by different spherical harmonics lighting environments. (still from Avatar, 2009, James Cameron, Weta Digital).

The spherical harmonics lighting is a rendering technique for calculating the incident illumination in computer-generated graphics. It utilizes spherical harmonics functions to represent the incident lighting term in the rendering equation. Figure 2.5 illustrates a scene from a movie illuminated by different spherical harmonics lighting environments.

The spherical harmonics functions are special functions, particularly interesting to represent functions defined on the surface of a sphere, such as the incident lighting in the rendering equation.

The spherical harmonics functions form an orthonormal basis. Any function in the function space defined by an orthonormal basis can be represented by a linear combination of the basis functions. For the one dimensional domain, the sinusoidal waves of the Fourier series utilized in Fourier analisys [44] are examples of orthonormal basis functions. This is the subject of the Signal Reconstruction subsection below.

The next subsections recapitulate the theory behind signal reconstruction, spherical harmonic functions, and presents the advantage of spherical harmonics functions to represent the incident lighting.

2.2.1 Signal Reconstruction

In the Signal processing field, basis functions, such as the spherical harmonics functions, can be understood as pieces of a signal that can be combined to reconstruct an approximation of the original signal. This reconstruction is accomplished by the sum of multiple pieces (basis functions) scaled by their contributions (scalar coefficients) to the original signal.

A particularly interesting subfamily of basis function is the orthonormal polynomial basis function. An orthonormal basis is defined by basis functions in which the integration of any two of them result in 0 or 1, as defined by equation 2.10.

$$\int_{-1}^{1} b'_{m}(x)b'_{n}(x)dx = \begin{cases} 0 & \text{for} m \neq n \\ 1 & \text{for } m = n \end{cases}$$
(2.10)

The main idea of the orthonormal basis function is that the contribution of each basis function does not overlap each other, similarly to the Fourier Analysis that breaks a signal into components sine-waves.

Given a signal f(x) and a set of orthonormal basis functions $B_i(x)$, it is possible to calculate the scalar coefficients C_i that approximate the original signal by integrating the signal by each basis function over the entire domain of the signal. This process of estimating the contribution of each basis function in the original signal is called projection. The projection process is shown in Equation 2.11.

$$C_i = \int f(x)B_i(x)dx \tag{2.11}$$

The reconstruction f'(x) of the function f(x) can be obtained by the sum of the basis functions $B_i(x)$ scaled by the associated constant C_i . This operation is shown in Equation 2.12.

$$f'(x) = \sum_{i} B_i(x)C_i \tag{2.12}$$

With an appropriate choice of basis functions, it is possible to represent an arbitrary function in a compact form. In this thesis dissertation, the spherical harmonics functions are chosen as basis functions to represent the incident lighting of the rendering equation.

The projection operation over the spherical harmonic basis is used to obtain the coefficients that represent a light probe image. The advantage of this approach is that only a set of scalar coefficients describes a complex light function (such as a light probe image).

This compact representation not only allows a speed up in the rendering process but also facilitate the learning process of the lighting estimation, described later on in this thesis.

2.2.2 Spherical Harmonics Functions

The Spherical Harmonics (SH) functions are a set of polynomial functions that define an orthonormal basis across the surface of a sphere [113]. The general spherical harmonics functions are defined for complex numbers. For lighting purposes, we use only real functions, so we consider only the real spherical harmonics functions (in this thesis, referred to as spherical harmonics functions).

The spherical harmonics functions take as parameters a point in the surface of a unit sphere in the spherical coordinates. We use a unit sphere (the radius has a fixed value of 1); thus, only the angular (ϕ, θ) terms are considered. The spherical harmonics are defined in bands parametrized with the numbers l and m. l is the band index (or degree in the polynomial functions notation), and it takes a positive integer. The m is the band order and takes a signed integer in the range [-l, l].

$$y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & \text{for } m > 0\\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta) & \text{for } m < 0\\ K_l^0 P_l^0(\cos\theta) & \text{for } m = 0 \end{cases}$$
(2.13)

K is a scaling function that normalizes the functions :

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$
(2.14)

 P_l^m is the associated Legendre polynomial, and the indices l and m are referred to as degree and order of the polynomial, respectively. The associated Legendre polynomials are recursively defined by the recurrency relations in the Equations 2.15 [113].

$$(l-m)P_l^m(x) = x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m$$
(2.15a)

$$P_m^m(x) = (-1)^m (2m-1)!! (1-x^2)^{m/2}$$
(2.15b)

$$P_{m+1}^m(x) = x(2m+1)P_m^m$$
 (2.15c)

$$P_0^0(x) = 1 \tag{2.15d}$$

The Equation 2.15a generates a higher degree polynomial based on the previous two degrees (l - 1 and l - 2) functions. The equation 2.15b requires no previous values so is suitable to raise the order m from the starting point P_0^0 , described in equation 2.15d. The equation 2.15c can be used to lift a degree l based on the value of a previous function P_{l-1}^m .

The process of evaluating the Legendre polynomial function P_l^m consist of generating P_0^m with the Equation 2.15b starting from equation 2.15d. Then use the Equation 2.15c to generate P_1^m and then iterate Equation 2.15a to generate P_l^m .

The SH functions in Equation 2.13 are defined in spherical coordinates. The relation between spherical and cartesian coordinate can be expressed as:

$$(x, y, z) - > (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$$
(2.16)

Table 2.1 shows the first three bands of SH functions converted from spherical to cartesian coordinates using the relations in Equation 2.16. The term r in the equations ensures the normalization of the vector:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.17}$$

A light distribution function projected in a three bands spherical harmonics basis results in a set of nine coefficients $c_{l,m}$, one coefficient for each SH function. The functions



Table 2.1: First 3 bands SH functions in cartesian coordinates. Source: [29].

Figure 2.6: Plot of the three bands spherical harmonic basis functions.

of the spherical harmonics basis provide spherical lobes that are used to express the environment's lighting contribution. Figure 2.6 is a plot of the nine functions of a three bands spherical harmonics basis. For the spherical harmonic lighting, the l = 0 function describe the ambient light term of the lighting. Positive values are represented in green and negative values represented in red.

We can use those coefficients in the rendering equation as the diffuse light contributions in a given normal direction n. The light contribution L(n) in the irradiance function can be expressed as :

$$L(n) = \sum_{l,m} c_{l,m} y_{l,m}(n)$$
(2.18)

For a normal vector in cartesian coordinates, we can use the spherical harmonics

Alg	orithm 1 Spherical harmonics evaluation algorithm
1:]	procedure SHRESOLVE $(normal, C) \triangleright C$ is the premultiplied coefficients array
2:	$result \leftarrow C[0]$
3:	$result \leftarrow result + C[1] * normal.y$
4:	$result \leftarrow result + C[2] * normal.z$
5:	$result \leftarrow result + C[3] * normal.x$
6:	$normalSqd \leftarrow normal * normal$
7:	$result \leftarrow result + C[4] * normal.x * normal.y$
8:	$result \leftarrow result + C[5] * normal.z * normal.y$
9:	$result \leftarrow result + C[6] * normalSq.z$
10:	$result \leftarrow result + C[7] * normal.x * normal.z$
11:	$result \leftarrow result + C[8] * (normalSq.x - normalSq.y)$
12:	return result
13: 6	end procedure

functions $y_{l,m}(x, y, z)$ of the Table 2.1. It is convenient to precalculate the constant terms of the functions $y_{l,m}(x, y, z)$ for usage in the shader step.

The code for the evaluation of the spherical harmonics is listed in Algorithm 1. The array C contains nine values resulting from the multiplication of the constant terms of the functions $y_{l,m}(x, y, z)$ and the coefficients $c_{l,m}$ of the SH light function.

2.2.3 Spherical Harmonics Light Probe

A practical approach to represent the real-world lighting in spherical harmonics coefficients is to capture a light probe image of the real ambient and project this image on a spherical harmonics basis.

The resulting projection is not an exact representation of the original light probe image, but an approximation that depends on how many functions forms the spherical harmonics basis.

It is possible to compare the approximation error by visually inspecting the reconstructed light probe image from a projection. This is accomplished by the following steps:

- 1. Capture a real-world light probe image.
- 2. Project the light probe image in a spherical harmonics basis.
- 3. Reconstruct the light probe image using the coefficients obtained in step 2.
- 4. Compare the light probe images of step 1 and 3.
Figure 2.7 illustrates the reconstructed light probe image as a result of projection operations using different SH order. Higher-order functions are used to faithfully represent the original signal (the light probe image), while low-order functions produce the low-frequency components of the light probe image.



Figure 2.7: Spherical harmonics light-probe image approximations.

A set of nine functions defines a third-order spherical harmonics basis, and it is considered sufficient to represent the lighting function for the diffuse environment lighting[92]. Figure 2.6 shows the plot of the nine spherical harmonics functions of a third-order spherical harmonics basis.



(a) 9 SH basis functions.



(b) 100 SH basis functions.



(c) Original light-probe image.

Figure 2.8: A virtual scene illuminated with different levels of spherical harmonics approximations.

Figure 2.8 depicts a scene rendered with a low-order spherical harmonic basis, a higher-order spherical harmonics light-probe, and the original light-probe image; the approximations yield a similar visual appearance when compared to the original light-probe image.

In this thesis, third-order spherical harmonics are utilized. However, the methods developed can be easily extended to a higher-order basis if necessary.

2.3 Machine Learning

Machine learning algorithms are algorithms that iteratively learn a function from the information present in a representation of the data [18], having an emphasis on the use of computers to statistically estimate complicated functions. This approach to algorithms allows solving problems that are too hard to solve with traditional programming.

The next subsections recapitulate the theory and concepts behind the machine learning algorithms employed in this thesis dissertation.

2.3.1 Learning Process

The learning process in a machine learning algorithm consists of providing examples to perform an experience that is said to solve a task. A performance measure evaluates the experience; the algorithm learns when the performance at the task increases [83]. The learning process is often referred to as the training phase.

An example, in the machine learning context, is a collection of features that have been measured from the target object or event to be learned. An example is typically represented as a vector $x \in \mathbb{R}^n$, where each feature of this example is a x_i of the vector. A collection of many examples forms a dataset. The experience described earlier is typically performed over the entire dataset. In this thesis, we also refer to an example as a sample of the dataset.

The supervised learning is a variant of the learning paradigm that tries to predict a target label from features of examples. A dataset for supervised learning algorithms contains examples with not only the features but also an associated label. The label is typically represented as a vector $y \in \mathbb{R}^m$. The learning process consists in estimating p(y|x) to predict y from x. Other variants of the learning paradigms do exist, such as unsupervised, semisupervised, multi-instance [57], and reinforcement learning [108, 3]. These other learning paradigms are out of the scope of this dissertation thesis.

The performance measure is a quantitative measure that evaluates the capabilities of the machine learning algorithms in a determined task. This measure directly depends on the task that the algorithm is solving and is actively employed to measure the performance of machine learning model; different metrics do exist, the choice of the metric utilized is a design decision that should correspond to the desired behavior for the specific task and the learning process.

A critical aspect of machine learning is to create models that perform well on new, previously unseen data; this ability is called generalization.

It is a common practice to separate the dataset into training, test, and validation sets. Ideally, these separated sets should come from the same distribution. The performance measures depend on the training, validation, and test set being disjoint subsets. Hence, they can share any example among them.

The training set is a portion of the data that is used during the training phase of the model. A performance measure called training error is calculated during this phase attesting how well the model is learning the training set.

The test set is a portion of the data that is used after the training of the model is completed; During the test phase, a measure called test error is calculated. The test error is a quantitative metric that provides the generalization of the trained model.

Most machine learning algorithms have settings that control the training process; these settings are referred to as hyperparameters. The learning algorithm does not alter the value of hyperparameters, but it is desirable to optimize these values during the training phases. The validation set allows tuning the hyperparameter during the training phase without compromising the generalization of the model.

2.3.2 Stochastic Gradient Descent

The learning process of a supervised learning algorithm involving the optimization by minimizing some function $f(\theta)$, $f : \mathbb{R}^n \to \mathbb{R}$ by altering the values of vector θ . The function $f(\theta)$ that is minimized is called the objective function. In machine learning, it is usually referred to as loss function, cost function, or error function.

The most common algorithm for minimization purposes is the Stochastic Gradient Descent (SGD) [7]. This optimizer is an extension of the gradient descent algorithm.

The gradient descent algorithm consists of finding θ' that minimize the loss function $f(\theta)$. The process is accomplished by moving θ in small ϵ steps, in the opposite direction of the gradient $\nabla_{\theta} f(\theta)$:

$$\theta' = \theta - \epsilon \nabla_{\theta} f(\theta), \qquad (2.19)$$

the ϵ is the learning rate, a positive scalar that determines the step size.

The gradient descent computations for a single step in a large dataset is prohibitive long. The SGD extends the gradient descent algorithm by estimating the gradient using a small set of samples (minibatch) of the train set. The minibatch of examples $\mathbb{B} = (x^1, ..., x^\eta)$ are drawn uniformly from the dataset. The batch-size η corresponds to a relatively small number of samples when compared to the total size of samples in the training set. The estimated gradient in the SGD becomes:

$$g(\theta) = \frac{1}{\eta} \nabla_{\theta} \sum_{i=1}^{\eta} f_{x^i}(\theta), \qquad (2.20)$$

we use the estimated gradient to calculate a step in the SGD optimizer as follow:

$$\theta' = \theta - \epsilon g(\theta), \tag{2.21}$$

Several improvements were made to the SGD algorithm. Those improvements include the AdaGrad [19], RMSProp [40], and Adam [53].

2.3.3 Feedforward Neural Network

The feedforward neural network is the fundamental algorithm that form the basis of many modern deep learning approaches. In the literature, it is also commonly called the multilayer perceptron (MLP).

The feedforward neural network is essentially a model that defines a mapping to approximate some function. For instance, in a classification problem, a classifier maps an input x to a category y. Hence a feedforward neural network approximates the function f' that satisfies the mapping y = f'(x) by learning the parameters θ of a function $f(x, \theta)$ that best approximates the function f'.

In this neural network, the information flows through the function, starting from the input x, following through the function f, and reaching the output y at the end. There are no connections in which the outputs of the model are fed back to itself. Hence the



Figure 2.9: Simple feedforward network representation. Each layer is composed of multiple units and all the units of a layer is connected with all the units of the previous layer.

name feedforward.

The feedforward neural network is composed of layers. These layers are different functions that are composed together, forming a direct acyclic graph that gives the architecture of the neural network. Figure 2.9 illustrates an example of a feedforward composed of three functions f^1 , f^2 , f^3 . The functions are connected in a chain, resulting in the function $f(x) = f^3(f^2(f^1(x)))$. In this neural network, the most inner function f^1 is the first layer of the network (also referred to as input layer), while the outer function f^3 is the output layer.

During the learning process, the training examples define the input layer x values and what the output layer should do to match the specified label y. The intermediate layers, between the input and the output layer, are called hidden layers. These layers are called hidden layers because the training data does not directly provide what calculations each layer should perform.

The neuroscience field historically inspired the creation of the neural network algorithms. The analogy comes from how the human neuron works, how they are connected, and how they exchange information through the synapsis in the brain. The layers of a neural network algorithm are typically represented by vectors where an element in this vector is analogous to a neuron. The elements work in parallel, where each element (neuron) is connected to elements from previous layers and compute its own activation value. The functions used to compute a layer mimics the synapsis that passes electrical signals from one neuron to another. The elements (or neurons) of a layer is called unit. Concerning units of hidden layers, they are referred to as hidden units.

A hidden unity h computes a function $f(\mathbf{x}, \theta)$ where the learned parameter θ consists



Figure 2.10: Hidden unity representation.

of the weights \mathbf{w} and the bias b as follows:

$$f(\mathbf{x}, \theta) = f(\mathbf{x}, \mathbf{w}, b) = \mathbf{x}^{\top} \mathbf{w} + b.$$
(2.22)

The previous equation utilizes linear operations. Therefore it describes a linear hidden-unity. As stated previously, the feedforward neural network operates by chaining multiple functions in the hidden layer.

The utility of linear hidden-unity is limited for a feedforward neural network considering that the resulting composition of multiple linear functions is equivalent to a single linear function; hence nothing is gained from using additional layers of linear units [1].

The expressive power of the feedforward neural network comes from composing multiple non-linear functions. The nonlinearity is consequence of using a non-linear activation function g in the hidden unit h, as follows:

$$h = g(\mathbf{x}^{\top}\mathbf{w} + b). \tag{2.23}$$

Figure 2.10 shows a graphical representation of the hidden-unity. Several non-linear functions can be used as activation functions of a hidden unit; the next section presents the activation functions utilized in this thesis.

2.3.4 Activation Functions

Historically, the logistic sigmoid was used as the activation function:

$$g_1(z) = \sigma(z) = \frac{1}{1 + e^{-x}},$$
(2.24)

this function is related to the hyperbolic tangent function [27], that is another commonly employed activation function:

$$g_2(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$
 (2.25)

The image of the hyperbolic tangent function is defined in (-1, 1), making it a useful activation function to apply a limit to the activated output of a given unit function. For a gradient-based learning method, the disadvantage of the sigmoid and the hyperbolic tangent functions are the saturation that occurs in most of its domain, hindering the learning process [27].

In a modern neural network, the most commonly utilized activation function is the rectifier linear unit (also referred to as ReLu) [46, 84, 26], it is a non-linear function defined as:

$$g_3(z) = \max(0, z). \tag{2.26}$$

The ReLu function is utilized due to its simple computation and the characteristic of preserving many of the properties that make linear models easy to optimize with gradient-based techniques [27].

Several other non-linear functions can be used as activation functions. It is desired that the activation function is differentiable in the entire functions' domain because of the gradient-based approach utilized in the neural networks. Although, functions that are not differentiable at specific values can be used, such as the ReLu function that is not differentiable for z = 0.

Generalizations of the ReLu functions were proposed by several authors in the literature [46, 73, 36, 28]; most of them have comparable performance when compared to the ReLu function [27].

2.3.5 Back-Propagation

The feedforward neural network utilizes the stochastic gradient descent algorithm, described in section 2.3.2, as the learning algorithm.

The stochastic gradient descent algorithm requires the computing of the gradient of the loss function; the back-propagation algorithm [96] is a numerical method that can be employed to computing the gradient of a function formed by composed functions with known derivatives, such as presents in the feedforward neural network.

The back-propagation algorithm computes the derivatives of the functions, using the chain rule of calculus with a highly efficient, specific, order of operations [27]. The algorithm is constructed using a computational graph language, defined by nodes that represents a variable, and a set of operations, that is a simple function of one or more variables. Goodfellow et al. [27] provides a formal definition of the back-propagation algorithm using the computational graph language.

2.3.6 Feature Learning

As described at the beginning of this chapter, the input of a machine learning algorithm is a dataset. The dataset is composed of examples, which in turn is a collection of features representative of the raw input data.

The features are properties and descriptors of domain-specific objects or events. Traditionally (before deep learning emerged), the features were manually engineered for a specific task, requiring much human effort to develop these descriptors from the raw data.

One of the main principles of the deep learning approach is to combine the idea of stacking multiple layers in the hidden layer to produce deep neural networks, with the idea of using the own model to learn the most desirable features that describe the data for a given task. The ability to learn the best representation of the input through the features is called feature learning. It also enables the transfer of the learned features between different tasks [120] and facilitates the learning process between different domains [85]. The next section describes the convolutional neural network, a deep learning neural network that is a specialized version of the feedforward network.

2.4 Convolutional Neural Network

The convolutional neural network [65] is a specialized machine learning algorithm for processing data in a structured grid-like topology. Examples of such data type include image (2D grid of pixels), and time-series (1D grid of data sampled in a regular time interval). A characteristic that this type of data shares in common is a strong local structure, where data that is spatially or temporally nearby are highly correlated.

The local correlations can be explored by extracting and combining local features before recognizing spatial or temporal objects [64]. The convolutional neural network applies this approach through the use of convolution operation.

Another critical characteristic of the convolutional neural networks is the ability to provide some level of shift and distortion invariance. This is accomplished with some properties such as local receptive fields (also referred to as sparse interactions), shared weights (parameters), and subsampling.

The convolution operation and its properties are presented in the following section of this thesis. Subsequently, a section discussing a subsampling strategy called pooling is presented.

2.4.1 Convolution Layer

From a machine learning perspective, the convolution operation implements three important properties: sparse interactions, parameter sharing, and equivariant [27]. A discussion about these properties and how they relate to the convolution operation is presented later in this section.

The convolution operation can be regarded as a weighted average of a function. The operation is defined in terms of an input function x(t), that represents measures sampled at time t, and a weighting function $w(\alpha)$, that represents the weight of the measure at age $w(\alpha)$; The convolution operation is denoted by an asterisc *, the output of such operation is called feature map and the operation is defined as follows:

$$s(t) = (x * w)(t) = \int x(\alpha)w(t - \alpha)d\alpha, \qquad (2.27)$$

as mentioned early, the output of the convolution operation is a weighted average, for that to happen, w should be a valid probability density function.

Taking into account that time is discretized, it is possible to write the discrete con-

volution as follows:

$$s(t) = (x * w)(t) = \sum_{\alpha = -\infty}^{\infty} x(\alpha)w(t - \alpha)$$
(2.28)

In machine learning terms, the input is a multidimensional array of data, while the kernel is a multidimensional array of parameters. For a two-dimensional case, with a two-dimensional input signal I and a two-dimensional kernel K; the discrete convolution operation over two axes is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(m,n) K(i-m,j-n), \qquad (2.29)$$

since the convolution operation is commutative, it can be written as :

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i-m,j-n)K(m,n).$$
(2.30)

0	0	1	1	0		
1	0	1	1	0		
0	2	0	0	1		
1	1	0	1	0		
0	0	1	1	0		
Input image I						

0	1	1		
-1	0	1		
1	-1	0		
kernel <i>K</i>				

0	0	1	1	0
1	0	1	1	0
0	2	0	0	1
1	1	0	1	0
0	0	1	1	0

(a) Starting position of the kernel matrix overlapping the image matrix.

	0	1	2		0	1	1		0	1	2		
$\overline{I \times K} =$	1	8	1	×	-1	1	1	=	-1	8	1	=	1
	0	2	0		1	-1	0		0	-2	0		

(b) Calculating the average of the element-wise multiplication of the image by the kernel matrix.

							•			_					
0	0	1	1	0	0	0	1	1	0		0	0	1	1	0
1	0	1	1	0	1	0	1	1	0		1	0	1	1	0
0	2	0	0	1	0	2	0	0	1		0	2	0	0	1
1	1	0	1	0	1	1	0	1	0		1	1	0	1	0
0	0	1	1	0	0	0	1	1	0		0	0	1	1	0

(c) Sliding the kernel matrix in a windowed form.

Figure 2.11: Steps to perform a convolution operation in an image.

In practice, this operation can be seen as a windowed image processing operation of

- (a) Positioning the kernel matrix overlapping the top-left corner of the image matrix Figure 2.11a;
- (b) Calculating the average of the element-wise multiplication of the kernel and image matrix values (Figure 2.11b);
- (c) Sliding the kernel matrix in a windowed form in the vertical or horizontal axis of the image (Figure 2.11c);
- (d) Repeat steps (b) and (c) for the entire image domain.

As mentioned early in this section, the convolution operation has some properties that are beneficial to machine learning algorithms:

Sparse connectivity refers to a scheme where each unity is sparsely connected to another one. This is achieved in a convolutional neural network by choosing a smaller kernel-size relative to the input image size. On the one hand, the input is usually in the order of thousands of pixels; on the other hand, the kernel matrix occupies in the order of tens of pixels. This scheme leads to fewer parameters being stored, reducing the memory requirements, and improving the model's statistical efficiency. The aforementioned scheme leads to computationally faster and efficient processing when compared to the feedforward neural network (Section 2.3.3), in which a unit is connected to every other unit in a previous layer, and a separated parameter for each connection is used. Figure 2.12 visually illustrates the difference between a sparse connectivity scheme and a fully connected scheme, like the ones employed in the feedforward neural network.

Parameter sharing is the idea of sharing the same parameter for more than one function in the model. The convolution operation utilizes the members of the kernel matrix at every position in the input image, which means that a convolution operation learns a set of parameters for the entire image. This approach differs from the feedforward neural network approach, which instead learns a parameter for each position of the image. The parameter sharing improves the computational aspect of the convolutional neural network algorithms by reducing the memory storage requirements of the model. Equivariance refers to the property of a function's output changes in the same way that the function's input changes when affected by a given transformation. The convolution operation has equivariance to (small) translations, meaning that it is possible to represent a signal transformed by a translation with the same convolution operation as the original, untransformed signal. The equivariance is a consequence of the parameter sharing approach adopted by the convolutional neural networks.

The convolutional layer is the layer that performs the most computationally intensive operations in the convolutional neural network. It produces a set of linear activations by performing several convolutions in parallel. It is usually followed by a nonlinear activation function.



(a) Sparse connectivity: a unity is connected to few outputs of the previous layer.



(b) Dense connectivity: a unity is connect to all outputs of the previous layer.

Figure 2.12: Sparse and dense connectivity.

2.4.2 Pooling Layer

The pooling function is a type of subsampling technique adopted in the convolutional neural network. The pooling operation consists of statistically summarize a certain portion of the data, usually a rectangular neighborhood area, by the use of some functions such

0	2	4	2			
2	0	2	4		1	3
1	5	5	2	(2 x 2) average pooling	5	2
12	2	0	1			

Figure 2.13: Average pooling operation

as the maximum value [118], the average, or the L^2 norm; of the neighborhood area. The work of Boureau et al. [8] provides more information on different pooling functions and their applicability.

The pooling operation has the effect of making the representation invariant to small translations and deformations of the input. In practice, it means that small translations in the input produce small differences in the pooled values. Figure 2.13 illustrates the pooling operation.

An additional benefit of the pooling operation is the spatial reduction of the layer output. This is accomplished by summarizing regions spaced by a constant k pixels apart. This constant k is referred to as stride of the polling operation. A pooling layer with stride k produces an output feature map with a spatial size approximately k-times smaller than the input data.

2.4.3 Residual Network Architecture

The ResNet architecture is based on a Residual Learning Framework [37]. As the depth of a deep neural network increases, the network begins to exhibit saturation in the accuracy. This saturation is rapidly followed by a degradation in the accuracy that leads to higher training error. This problem is called the Degradation Problem.

In Deep Learning, the Degradation Problem arises as a consequence of the solvers having difficult in approximating identity mappings from a set of nonlinear layers. The insertion of a preconditioning residual learning function can help the solver to deal with the degradation problem. The Residual Learning Framework [37] introduces residual learning functions that are closer to identity mappings. The residual learning functions are modeled by shortcuts between the layers in the network.



Figure 2.14: The basic building block of a Residual Network. The building block has a shortcut connection and follows the bottleneck design.

The identity shortcut connects two different layers and produces new output feature maps by an element-wise addition operation. There are two or more layers between the shortcut connected layers.

The projection shortcut connection is similar to the identity shortcut, but an additional linear projection is applied to match the dimensions of the connected layers.

The construction of a CNN that follows the ResNet architecture can be accomplished by the introduction of residual blocks. A Residual block has the following characteristics:

- A shortcut connection is made to connect the input of the first layer in the building block to the output of the last layer in the building block. The identity shortcut connection is used when the dimension of the input and output are the same. Otherwise, the projection shortcut connection is used, in this case, a 1 × 1 convolution with a stride 2 is used to match the dimension in the feature maps.
- A bottleneck design is used to decrease the required training time. In the ResNet architecture, this design consists in replacing two 3 × 3 convolution layers by a stack of three layers following the convention: 1 × 1, 3 × 3, and 1 × 1 convolution layers. The 1 × 1 layers are used to create a bottleneck by reducing the input and output dimensions in the 3 × 3 layer. Due to the matching dimensions in the first and last layers of the building block, the bottleneck design allows an identity shortcut to be used instead of a projection shortcut. The identity shortcuts result in smaller model size and consequently lower time complexity.

Figure 2.14 illustrate the basic bottleneck residual building block for a building block with an input of n feature maps.



Figure 2.15: Residual Network Design : Stacking Residual building blocks.

The design of a ResNet CNN consist of stacking the building blocks following two rules:

- 1. Consecutive building blocks that have the same feature map dimension size must maintain the same number of filters in the 3×3 convolution layer.
- 2. The time complexity per layer is preserved by doubling the number of filters every time that a feature map dimension size is halved.

Figure 2.15 illustrates a valid design for residual building block stackings in the ResNet architecture. The element-wise addition and the ReLu functions present in the output end of the shortcut connection were omitted for a better visualization.

In this example, the first and second residual blocks output n feature maps with the same dimension size. Therefore, according to the first rule, they must have the same number of filters in the 3×3 convolution layer. The second and third building block have different feature map dimension size, so a 1×1 convolution layer with a stride of 2 was used to halve the feature map dimension. According to the second rule, the number of filters in the layers of the third residual block were doubled (resulting in 2n filters). Note that the second rule is also applied to the 1×1 convolution layer responsible for downsampling the output of the second building block.

2.4.4 Dense Convolutional Network Architecture

The DenseNet is based on the Densely Connected Convolutional Network [45]. The Dense Convolutional Network explores the observation that shorter connections between layers close to the input and layers close to the output lead to more accurate and efficient models. This observation was observed in previous architectures, including the Residual Network presented in the previous section.



Figure 2.16: The basic building block of a Dense Convolutional Network. The dense block has a densely connected scheme in which layers are connected to all the preceding layers within the dense block.

The Dense Convolutional Network introduces a layer connection scheme where any given layer is connected to all the features-maps of the preceding layers. This connection scheme increases the number of direct connections of a network with L layers from L connections to L(L + 1)/2 direct connections. The connection scheme of the DenseNet leads to significant improvements over the previously proposed architectures, leading to high performance and requiring less computation [45].

The construction of a CNN that follows the DenseNet architecture can be accomplished by the introduction of Dense blocks. Figure 2.16 illustrate the dense building block. The dense connection scheme is a key characteristic in the DenseNet architecture. A Dense block has the following characteristics:

- Connections are made to connect the input of any given layer in the building block to the output of all the preceding layers within the building block. The connection is made through the use of a composite function: a batch normalization layer activated by a ReLu function, followed by a 3 × 3 convolution layer. The multiple inputs of a layer are concatenated into a single tensor to facilitate the implementation.
- The pooling operation of the network is performed by a batch normalization layer accompanied by a 1 × 1 convolution layer and followed by a 2 × 2 average pooling layer. This pooling operation is present in the transition separating two successive dense blocks. Thus, this set of operations is called the transition layer.
- The hyperparameter growth rate k, controls the number of feature-maps a layer produces. Due to the densely connected scheme, each l^{th} layer of the network has

 $k_0 + k \times (l-1)$ input feature-maps, where k_0 is the number of channels in the input layer.

- A bottleneck design is used to decrease the number of feature-maps in the input of a layer. This is accomplished by stacking a 1×1 convolution layer before each 3×3 convolution layer of the dense block.
- The 1 × 1 and 3 × 3 convolution layers of the dense block are preceded by a batch normalization operation and pre-activated by a ReLu activation function.



Figure 2.17: Dense Network Design : Stacking dense building blocks.

The design of a DenseNet CNN consist of stacking the building blocks in which consecutive building blocks are connected by a transition layer that performs the downsampling of the features in the network. Figure 2.17 illustrates an example of a DenseNet architecture design composed of multiple dense blocks.

2.4.5 Neural Architecture Search

The NASNet is a convolutional neural network based on the Neural Architecture Search (NAS) architecture [120]. The Neural Architecture Search defines a method in which the overall architecture of the neural network is predefined, but the blocks that composite the architecture are searched thought the use of a reinforcement learning search method.

There are two building blocks in the NASnet architecture:

- The Normal Cell is a block that output a feature map of the same dimension than its input.
- The Reduction Cell is a block that output a feature map with a dimension reduced by a factor of two.

The combination of several operations composes the layers of the Normal and Reduction Cells in the NASnet architecture. Convolution and pooling layers with different kernel sizes are the operations utilized in the NASNet architecture.



Figure 2.18: The NASNet overall architecture. The NASNet is composed of normal and reduction Cells. The number of repetions n of those cells is a parameter to scale the network layers count.

The construction of a CNN that follows the NASNet architecture can be accomplished by stacking the normal and reduction cells. The CNN can be scaled by repeatedly stacking normal, and reduction cells by a free parameter n.

A particular implementation of the NASNet architecture utilized in this thesis is the NASNet-A [120]. This network is the result of a search process that spans over four days of processing using a massive amount of 500 GPUs. The search was based on the

performance of the architecture for the CIFAR-10 dataset. The normal cell and the reduction cell found in this process are shown in the Figure 2.19 and Figure 2.20.



Figure 2.19: NASNet-A normal cell.

The operations that compose the normal and reduction cells of the NASNet-A network are drawn from a set of the following operations:

- **identity map** operations that uses a identity function in the input resulting in a shortcut connection.
- convolution operations using several variants of filters in respect to the kernel size, the kernel size varying from 1×1 to 7×7 .



Figure 2.20: NASNet-A reduction cell.

pooling operations using several variants concerning the pooling function and the kernel size: max and average pooling functions with kernel size varying from 1×1 to 7×7 .

concatenation and addition operations that combine previous layers data.

The NASNet-A achieved state-of-art performance on CIFAR-10 image classification, ImageNet image classification, and COCO object detection. Furthermore, the number of learning parameters of the NASNet-A model is considerably smaller than comparable models [120].

Chapter 3

Lighting Estimation Related Works

This chapter presents a discussion about the state-of-art on lighting estimation methods. Several approaches to lighting estimation are discussed and categorized into groups. Later in this chapter, a listing of the most relevant existing works are displayed in a table format.

3.1 Overview

Estimating the lighting settings of a scene is an essential process that is not limited to the mixed reality context but also employed for a wide variety of applications in the computer graphics domain, especially for applications that use real-world data to generate 3D virtual models. Some applications examples are special-effects for cinema, 3D scene reconstruction, environment planning, and designing.

This thesis is focused on the lighting estimation methods that are suitable for mixed reality applications. A key characteristic of this kind of application is the low latency response between the real and virtual worlds (changes in the real-world should reflect in changes in the virtual world with minimum delay). The low latency response is achieved by a real-time lighting estimation method. In other words, it means that the lighting estimation should be fast enough to be calculated in real-time by the applications.

We categorize the lighting estimation techniques concerning the required resources, method constraints, and method specializations.

Section 3.2 describes device-based light probe methods. The methods described in this section make use of physical devices to measure real-world lighting.

Section 3.3 describes methods that assume prior knowledge of the scene geometry or

estimates the geometry of the scene through the use of depth-cameras.

Section 3.4 describes methods that are only suitable for specific applications and are not generalizable to mixed reality applications.

Section 3.5 describes methods that jointly estimates surface reflectance and lighting.

Section 3.6 makes a summary of the presented lighting estimation methods and compares the advantages and disadvantages of those methods for mixed reality applications.

3.2 Device-based light probe

The lighting estimation of a real-world scene can be accomplished by the use of physical objects that measure the scene's radiance information. These objects have specific characteristics that allow probing the environment light settings. Thus they are referred to as light-probes.

The device-based techniques, presented in this section, comprehend methods that make use of devices that act as light probes or directly measures the lighting condition of the scene.

The image-based rendering method introduced by Debevec et al. [16, 15, 14] takes advantage of a physical light-probe placed directly into the real-world scene to measure scene radiance. The object that they utilized as light-probe is a mirror ball, this is a smart choice considering that the highly reflective surface of the sphere allows capturing the radiance of the environment, while the spherical geometry provides a high field of view of the environment. They capture high dynamic pictures of the mirror ball and assembles the pictures to create a light probe image of the environment. Later they use the light probe image to render a virtual scene.

The process of capturing the high dynamic pictures of the light-probe is laborious (several pictures of the device with different exposure levels are necessary) and requires a proper setup of the scene, rendering impractical for mixed reality applications.

Succeeding works tried to minimize the efforts regarding the capturing process of the pictures by designing light-probes that can estimate the environment light with low dynamic range pictures [10, 2]. The work of Calian et al. [10] proposes the design of a specialized light-probe that is capable of capturing the shading of a scene; the light-probe design is created by piecewise-constant basis functions. They produce the physical lightprobe by 3D printing, the light-probe is positioned in the environment, and a 2D printed marker registers its position. The process of acquiring the shading of the scene demands pictures of the light probe in different views. They acquire those pictures by manually rotating the camera around the light-probe. The shading samples are captured from the faces of the probe.

Another approach for acquiring the radiance of scene consists of acquiring pictures of the environment with a wide-angle camera. This type of camera is usually equipped with a fish-eye lens that is capable of acquiring the entire environment with few pictures [110, 50, 55]. The straightforward advantage of this approach is the low number of pictures required to create a light probe image of the environment. Usually, two pictures are sufficient. The clear drawback of this approach for mixed reality applications is the requirement of additional hardware (wide-angle cameras).

A recent approach to lighting estimation employs a single-exposure picture containing several physical light-probes with different surface reflectance [66]. Following this approach, LeGendre et al. utilize a capture apparatus that introduce three spheres with different reflectance into the real-world scene. The spheres provide different lighting cues from a single low dynamic range image. They use a deep neural network to regress the light-probe image of the environment. The deep neural network is trained by comparing the picture of the real spheres with 3D spheres rendered with the estimated environment lighting. This approach requires capturing a high number of examples as the training data for a variety of environment and lighting settings. In their work, this process is handled by collecting video-frames using a camera placed in a capture rig that arranges the three spheres in a fixed position, taking the lower portion of the video frame. The laborintensive capture process of the training data is considered a downside of this approach. The methods developed in this thesis aims to overcome that labor-intensive aspect of the training data capturing process by producing synthetic examples that populate our training data (Section 4.4).

A common issue that limits the usage of physical light-probes for mixed reality applications is the requirement of the light-probe be physically located and visible in the real scene. This limitation renders the usage of the previous methods unpractical for in the wild mixed reality applications.

3.3 Know geometry

This section assembles lighting estimation approaches that rely on the prior knowledge of the real-world environment geometry or the prior knowledge concerning the geometry of a light-probe object that is physically located in the real-world environment.

Geometry and surface reflectance knowledge are convenient assumptions when solving inverse rendering techniques, particularly for inverse lighting methods [79]. The methods described below explore those assumptions to estimate the environment lighting.

A typical approach to obtain the geometry of a real-world object consists of reconstructing the object using depth-aware images. The depth is captured by employing a depth camera (depth cameras that combines depth and regular RGB data are referred to as RGB-D cameras). The depth camera provides the spatial depth distance for any pixel in the captured image. Although popularized by the Kinect [103], depth cameras are not currently employed in the virtual reality and mixed reality HMD hardware.

Some methods do not assume previous knowledge of the scene geometry but utilize a rough estimation of the depth or surface normals of the environment [114, 74, 116, 94, 80, 30, 31, 88]. These methods usually combine photometric registration techniques to estimate discrete light-source positions based on lighting cues presented in the virtual 3D reconstruction of the scene. The method presented by Knecht et al. [55], described in the previous section, is an example of such a method that implements a depth-based reconstruction of the real-world scene to estimate a set of point light-source positions in the real-world environment.

Following the idea of using the scene's geometry to estimate lighting, Boom et al. [6] proposed a method to estimate an individual point-light source based on the geometry of the scene. An RGB-D camera is used to capture the geometry of the scene. They use an image segmentation to find regions of the image with similar albedo; this segmentation provides the necessary information of the objects' material in the scene. The geometry and material information (reflectance) allow the 3D reconstruction of the original scene under different lighting configurations. They search for the best position of the light source by a minimization process between the reconstructed scene and the captured image of the real scene. Jiddi et al. [48] proposed a similar method for lighting estimation that could handle multiple point-light sources in the scene. The multiple point-light sources is estimation is accomplished by analyzing the specular reflections in the surface of objects located in the scene.

Mandl et al. [75] present a method that estimates a light probe image of the real-world environment. The method can use any object physically located in the scene as a lightprobe. Prior knowledge of the light-probe geometry and material (reflectance) is required. The method explores the geometry and reflectance knowledge to create a training dataset for a convolutional neural network. They generate the training examples by rendering the lighting probe with varying camera poses. Their method relies on numerous deep neural networks, one for each camera pose. They apply a pose estimation algorithm [54] to determine the camera pose in the real-world scene and use this information to select which previously trained neural-network to use in real-time.

Weber et al. [112] propose a latent-space representation of the environment lighting. The light probe image is encoded in a 128-dimensional vector through the usage of a convolutional autoencoder. The lighting estimation is performed by a convolutional neural network that maps the appearance of an object to the proposed latent space. This neural network takes as input both an LDR image and a normal map of the object and outputs a 128-dimensional vector. This output vector is a compressed representation of the lightprobe image. The training dataset for the lighting estimation network is composed of synthetic examples. The examples were created by rendering 3D models with known geometry and reflectance; the 3D models were randomly rotated to produce random poses. They utilize an IBL rendering method with light-probe images derived from an HDR indoor panorama dataset [22].

The lighting estimation from real data requires an object with known geometry and reflectance. A Kinect is employed to capture the light-probe object pose in the scene using an RGBD pose estimation algorithm [23]. They render a virtual version of the light-probe to acquire the normal map and a mask of the object. This mask of the object assists in further processing the RGB picture taken by the Kinect, effectively removing the background pixels of the image. Finally, they concatenate the RGB image with the normal map of the object and feed it to the trained neural network. The outcome of the method is a light probe image of the environment.

The methods described above require computationally expensive processing steps and manual processes such as the 3D scanning of the light-probe object. Furthermore, it requires multiple renderings of a specific light-probe object to create a suitable training dataset. Hence, the trained network only works for that specific light-probe, and the training process should be repeated when another light-probe object is employed in the scene. This process is potentially unpractical, especially concerning Mandl et al. [75] method that requires the training of multiple deep neural networks for a single lightprobe object.

3.4 Domain-Specific Methods

Lighting estimation is a task that arises from the problems of multiple domains. In this section, we explore methods that are tailored to estimate environment lighting in particular domains, and that cannot be generalized to a wide range of applications due to restrictions in the input data or the usage of domain-specific lighting models.

Several works concentrate on the lighting estimation for images of human faces [119]. Relighting of portraits is a problem that requires the lighting estimation of the environment where the picture was taken. Usually, the 3D morphable model of faces [4] is utilized to perform an analysis-by-synthesis capable of estimating the environment lighting [98, 13, 99]. Building on the morphable model, Egger et al. [20] present an occlusion-aware face model adaptation that deals with occlusion and complex illumination conditions, allowing the estimation of more sophisticated environment lighting settings. Recently, Sun et al. [107] presented an alternative methodology for portrait relighting that employs a convolutional neural network to estimates the environment lighting. These methods rely on images of human faces in portrait poses to estimate the environment lighting, making them domain-specific methods and a limit factor for mixed reality applications.

The work of Calian *et al.* [9] presents a method for lighting estimation that utilizes a person's face for a task unrelated to portrait relighting. Their method employs the person's face as an outdoor light-probe. Outdoor lighting estimation is another domainspecific task that was explored by several authors [117, 41, 59, 60]. They tackle the problem by fitting specific parametric outdoor lighting models [43, 62, 87, 34] that take into account sun and skylight properties. An example of such methods is the approach adopted by LaLonde et al. [61]. Their method estimates the parameters of a sky lighting model [43] by analyzing shading and shadow cues on a low dynamic range image. The method presented by Hold et al. [42] also focus on the outdoor environment, but uses a different approach in which a convolutional neural network infers the parameters of the sky lighting model.

Considering that our goal is to estimate lighting for an arbitrary environment, we do not further investigate outdoor-specific lighting estimation methods.

Some works [22, 104, 24] concentrated on the lighting estimation specific to indoor

48

environments. Those methods do not generalize to the environment lighting estimation at outdoor environments or environments with uncommon lighting scenarios due to restrictions of the light source models employed. Gardner et al. [22] propose a lighting estimation specific to indoor environments; the method makes use of a CNN that estimates a coarse light-probe image of the scene; the method achieves good results for relights of indoor environments. However, it relies on an intermediate process that makes use of a light classifier to annotate the location of light sources; this process is a possible limiting factor for the applicability of the method in outdoor scenes and non-conventional light sources (for example, indirect light sources resulting from global illumination effects). The method developed by Song and Funkerhouser [104] estimates the environment lighting in a localized point of an indoor scene by decomposing the problem in subtasks: geometry estimation, warped panorama completion, and LDR to HDR estimation. Decomposing the problem in simpler subtasks helped the learning process, producing higher quality estimations when compared to Gardner et al. [22] approach. The method developed by Garon et al. [24] estimates local lighting of the indoor environment by combining local and global lighting; this is accomplished by providing both the global image and a local patch of the image to a two-path neural network. The training process of this network requires light-probes with depth maps of the scene. To circumvent the laborious process of capturing the light probes and depth maps on real scenes, they make use of synthetic data to generate the required training data.

The usage of human eyes as light probes is another approach that has been investigated [109, 86]. The human eye has a known geometry that can be approximated by a sphere. Furthermore, the high specular reflectance makes the eye a suitable probe for lighting estimation [111]. The downside of this approach is the reduced pixel resolution. Furthermore, a human face should be visible in the image, posing as a limitation of the method. This limiting assumption is hard to circumvent in a mixed-reality egocentric view perspective.

3.5 Surface reflectance and lighting

Since the environment lighting has a significant influence on the surface reflectance, recovering surface reflectance is a task correlated to lighting estimation. Under controlled environment, surface reflectance methods make use of planned lighting settings and explore the known lighting to infer the material properties [17, 70, 11, 32, 25]. The factorization of a single image into a product of reflectance image and shading image is a problem known as intrinsic image decomposition [68]. Several works [69, 52, 67] address the real-time intrinsic image decomposition problem to estimate surface material. Although it is not the main contribution of those papers, given the material and geometry of an object, it is possible to determine the lighting by projecting the estimated material image to a spherical environment map [81].

A typical context for surface reflectance estimation is to estimate the surface reflectance under unknown environment lighting. Most methods estimate the reflectance and environment lighting simultaneously, using either the geometry of a 3D scanned object [11, 71] or depth sensors [48, 72]. Thus, those methods share the limitations present in the geometry-based lighting estimation methods.

3.6 Related Works Summary

The related works presented in this chapter share constraints that limit the usage of those methods in mixed reality applications. We summarize those methods advantages, constraints, and limitations below :

The methods described in Section 3.2 produce plausible environment lighting estimations but poses some hard to circumvent limitations. The most notable limitation is the necessity of a light-probe object being physically present and visible in the scene. Moreover, some of the methods presented make use of wide-angle cameras or complicated setups such as a properly calibrated camera placement to capture high dynamic range pictures of the environment. The obtrusive light-probe object located in the scene and the usage of specialized types of equipment restricts the usage of the methods to previously prepared environments. Therefore, rendering those methods unpractical for general, in the wild, mixed reality applications.

The methods described in Section 3.3 utilize depth sensors (or RGB-D cameras) to capture scene geometry information. Although the recent advancements in the depth sensor technologies and the low-cost availability, the current generation of virtual reality and mixed reality hardware do not provide such capabilities; moreover, those techniques require the scan of the complete scene before a reasonable lighting estimation becomes possible. Furthermore, depth sensors are inaccurate and produce noisy results when utilizes in environments with high infra-red light incidence. This characteristic compromises the efficiency of the previous methods in outdoor environments. The methods described in Section 3.4 are environment lighting estimation methods tailored to work in specific domains, and due to technical reasons, do not generalize to a generic mixed reality environment.

The methods described in Section 3.5 takes into account the recovery of the surface reflectance of objects and jointly estimates the environment lighting of the scene. Those methods employ intrinsic image decomposition techniques that factorize an image in the reflectance image and shading image. This factorization process is related to visual cues about the material, geometry, and lighting of the environment present in the original image. To jointly estimates material and lighting, those methods utilize the prior knowledge of the scene's geometry, sharing the same constraints as the methods presented in Section 3.3.

Different from the previous related works, the methods developed in this thesis aims to be practical and effortless for the end-user. The methods presented in this thesis does not rely on any special equipment or prior knowledge of the scene.

In the next chapters, three environment lighting estimation methods developed for this thesis are presented. The developed methods use an RGB image as input and output the environment lighting estimation. Our methods work on both the indoor and outdoor environments and do not require any particular scene setup.

The first two methods assumed that the user's hands are visible in an egocentric view of a mixed reality environment.

The first method, the Point Light Source Estimator (PLSE), estimates the position of a single point light-source responsible for the main lighting contribution in the scene.

The second method, the Spherical Harmonics Light Probe Estimator (SHLPE), estimates the parameters (coefficients) of a spherical harmonics basis functions that are used as a light-probe of the environment.

The third method, the Environment Light Probe Estimator, generalizes the SHLPE method by allowing the lighting estimation in any given scene, lifting the constraints of the user's hands be visible in the scene.

The table 3.1 summarizes the most relevant discussed methods for lighting estimation. The methods developed in this thesis, listed in the first three rows of the table, are the only ones that do not require any special equipment or laborious scene setup and are capable of estimate the lighting in both indoor and outdoor environments.

Method	Input data	Output format		tions		
		RGB-D or fish-eye	Physical light-probe	Indoor env. only	Outdoor env. only	
ELP Estimator	RGB image HMD rotatition	27 spherical harmonics lighting coefficients				
SHLP estimator [76]	RGB image with user's hands	9 spherical harmonics lighting coefficients		×		
PLS estimator [77]	RGB image with user's hands	Single point light source position		×		
Gardner et al. [22]	RGB Image	HDR environment mapping	×	×	×	
Legendre et al. [66]	RGB Image	Light-probe image		×		
Garon et al. [24]	RGB Image	Spherical harmonics lighting coefficients			×	
Song et al. [104]	RGB Image and Scene geometry	light-probe image mapping	×		×	
LaLonde et al. [61]	RGB image	Sky lighting model parameters				×
Hold et al. [42]	RGB image	Sky lighting model parameters				×
Zhang et al. [117]	RGB image	Sky lighting model parameters				×
Calian et al. [10]	RGB image containing the light-probe	Shading parameters		×		
Debevec et al. [16]	HDR images of the light-probe	Light-probe image		×		
Mandl et al. [75]	RGB image and light-probe geometry and material.	9 spherical harmonics lighting coefficients	×	×		
Trummer et al. [94]	light-probe geometry and material	Spherical harmonics lighting coefficients	×	×		
Choe and Shim [12]	RGB-D image	light-probe image	×	×		
Boom et al. [6]	RGB-D image	Single point light source position	×	×		
Jidi et al. [48]	RGB-D image	Multiple point light source position	×	×		
Knetch et al. [55]	Wide-angle image and RGB-D image	Point light source position	×			
Pessoa et al. [88]	Wide-angle image and light-probe geometry	Light-probe image	×	×		

Table 3.1: Lighting estimation methods comparison

Chapter 4

Lighting Estimation on Mixed Reality

The lighting estimation is an important task to achieve consistent lighting in mixed reality applications. The focus of this thesis is the development of lighting estimation methods that possesses the following characteristics :

- The algorithm should receive an image in an egocentric view (also referred to as the first-person view)
- The algorithm should execute in real-time, performing estimation fast enough to be executed during the application runtime.
- The algorithm should not depend on specialized hardware.
- The algorithm should not depend on manual steps that require the user's intervention (camera calibrations and scene setup).

Those characteristics make the use of the lighting estimation method practical and straightforward at mixed reality applications. The method developed in this thesis aims to achieve such characteristics employing deep learning approaches.

The deep learning models employed, particularly convolutional neural networks, are models suitable for 2D images, have an inference time of few milliseconds, and does not depend on user's manual intervention.

An appropriate choice of a model that represents the environment lighting should be made, Section 4.1 discuss the lighting representations utilized in the lighting estimation methods developed in this thesis. During the development of this thesis, we assume that it is possible to estimate the environment lighting using a human hand as a light-probe. The reasoning and implications of this assumption are discussed in Section 4.2.

The lighting estimation methods developed in this thesis use a common framework to create consistent lighting in mixed reality applications. The framework describes the steps to ensure consistent lighting between the real and virtual worlds by adjusting the virtual lights to match the real-world lighting. Real-world lighting estimation is a crucial part of this framework. The mixed reality lighting framework is discussed in Section 4.3.

4.1 Light Representation

The calculation of physically accurate lighting models that obey physical theoretical basis (for example, physically correct light transport and the law of conservation of energy) requires a substantial computational effort and time. The heavy computations are not feasible in real-time applications where the usual target for the frame rendering time is in the range of 16 to 33 ms [100]. For this reason, real-time applications rely on heavily approximated models.

Light-source models are employed to represent the lighting setup in a virtual scene. The light-source models are heuristics that approximate how real light-source works. Those heuristics can be combined with empirical reflectance models [89, 5], resulting in a computationally efficient and extensively used method for interactive and real-time lighting and shading.

The empirical light-source models are phenomenological models [58] that are empirically built to visually represents most of the significant optical phenomena of real light-sources.

The point light-source model is an empirical light-source model that gives an equal amount of light in all directions. The point light-source model is defined by its position in space, the lighting color, the intensity, and the attenuation function of the light.

The point light position defines the spatial position of the light-source in world coordinates and is usually stored in a vector $l_p = (x, y, z, 1)$. The lighting color describes the exitance radiance frequency distribution of the light source and is stored in a vector $l_c = (r, g, b, 1)$. The intensity is a scalar l_i that modulates the total contribution of this light source in the lighting transport calculations. The attenuation function $l_a(d)$ is a heuristic function that modifies the intensity of the lighting; this function simulates the dimming effect that occurs in real light-sources when the spatial distance d between the light source and the hitting surface increases.

Due to its mathematical and computational simplicity, the point light-source model is one of the light representation explored in this thesis methods.

Another efficient way to represent the environment lighting is through the usage of image-based lighting techniques (Section 2.1). Those techniques make use of measured environment radiance stored in light-probe images that are capable of representing the incident illumination conditions in the entire environment.

The light-probe image pixels stores the radiance of the scene. The disadvantage of this light-source representation is the required memory to store and process these images. Since the radiance spans over a wide range of real values, it is not viable to use the traditional 24 bits per pixel storage scheme (RGB), which stores each color channel in 8 bits integers. Thus, for light-probe images, a high dynamic range image format should be used; a logical approach is to store the light-probe image in a float image format, in which each color component of a pixel is stored in a 32 bits float format, taking considerable storage space. For comparison, an image with 2048 x 4096 pixels stored in the traditional 24 bits per pixel, without any compression, would occupy approximately 25 MB of storage, while an HDR image of the same spatial resolution, stored in a 32 bits float format would take approximately 101 MB of memory space.

Spherical harmonics lighting (Section 2.1) is employed to alleviate the high usage of memory and computational processing power of the light-probe images. A light-probe image approximated by a third-degree spherical harmonics basis is stored in three vectors with nine components each, resulting in 27 floats, thus occupying only 108 bytes of memory.

The usage of spherical harmonics lighting is not beneficial only due to the low memory required. The optimization process in the training of a neural network could benefit from this compact representation as well. It is easier to fit a function with dozens of parameters than a function with millions of parameters.

The spherical harmonics lighting provides a compact representation of lighting that can model complex area-light effects that are not feasible with the point-light source model. The light-probe image expressed in spherical harmonics functions is another lightsource model explored in this thesis.

4.2 Human Hands Light-Probe Hypothesis

In mixed-reality, the user interacts with the ambient in many ways (through physical controllers, sensor-based interactions, virtual tracking interactions). There is an ongoing effort to make interactions more natural. A possible approach is to use the user's hands as a controller, allowing the user to interact with the virtual world naturally, this approach is a logical step due to the behavior of the humans' interactions in the real world. This approach explores the reasonable assumption that the user's hands are always available and idle in a regular mixed reality application.

Some of the work developed in this thesis explores the hand-based interaction models. In this interaction model, the user's hands are actively performing actions such as grabbing, holding, and pushing virtual and real objects. For this reason, it is reasonable to assume that the user's hands are visible in the view-frustum of the rendered scene, and also visible by the HMD camera. This assumption holds when the user is performing some interactions with the environment. Due to the interactive nature of mixed reality applications, it is possible to consider that an image of the user's hands is available at a reasonable amount of time for a mixed reality application.

Some methods developed in this thesis use the hypothesis that the user's hands contain sufficient information about the environment lighting and thus can be used as light-probes of the real-world environment.

The diffusive nature of human skin can pose a challenging aspect of the usage of human hands as light-probes. However, several works estimate environment lighting concerning human faces acting as light-probes (see Section 3.4 of the related works). The success of those techniques makes the hand based light-probe hypothesis stronger.

4.3 Mixed Reality Consistent Lighting Framework

The consistent lighting between the real and virtual world is an indispensable feature to create compelling and visually pleasant mixed reality applications. There is evidence that increased visual realism leads to a higher user's presence in immersive virtual environments [101].

The mixed reality framework developed in this thesis and presented in this section alleviates the illumination mismatch problem (introduced in Section 1.2) by employing an adaptive lighting scheme for mixed reality applications.



Figure 4.1: General mixed reality framework.

The framework is focused on lighting estimation models. It is designed to recovery the real-world lighting and applies the recovered lighting into the virtual objects of the mixed reality application. The general approach of the framework consists of a continuous processing of the environment lighting, making use of the Algorithm 2 to update the mixed reality environment.

Alg	Algorithm 2 Mixed reality framework frame update algorithm					
1:	1: procedure Framework					
2:	Take the input image of the real-world scene					
3:	Process the input image into a format suitable for lighting estimation					
4:	Use a lighting estimation model to estimates the real-world lighting settings					
5:	Give the estimated lighting to the rendering engine					
6:	Apply the estimated lighting to the virtual world					
7:	7: end procedure					

This process should be continuously repeated to ensure that any change in the lighting of the real-world would be reflected in the virtual world, ensuring consistent lighting. Figure 4.1 is a flowchart that illustrates the general steps of the mixed reality framework.

The hand-based specialized version of this framework considers the hand-based light probe hypothesis (Section 4.2). Additional steps are performed to check if the user's hands are visible in the current frame, and also give to the game engine an image containing only the user's hands (the background is extracted from the image).

The segmentation of the user's hands from the image is essential for the learning process of the developed lighting estimation models, and also have benefits for some
classes of mixed reality applications. A particular example of such applications is the hand-based augmented virtuality application (introduced in Section 1.1).

Hand-based augmented virtuality applications can benefit from the technological maturity and low-cost of virtual-reality HMDs allowing hand-based interactions with the visual introduction of the user's real hands into virtual worlds without extra hardware or costs.

To accomplish such a task, a stereo camera (such as the cameras present in the HTC Vive Pro [63]) captures an egocentric picture of the environment. Applying the hand based mixed reality framework with this picture as the source input results in a composite image of the rendered virtual world and the user's hands.

Figure 4.2 is a flowchart that illustrates the overview of the hand-based mixed reality framework. Details of each step of the framework are discussed in the following sections.



Figure 4.2: Hand-based mixed reality framework overview: A camera feed acquires the real environment, the input picture is an RGB color image, a hand visibility test is performed to check if the hand is visible in the image to proceed with the lighting estimation. A preprocessing step is performed to separate the hands from the background pixels of the real environment; the resulting segmented image is fed to both the Lighting Estimator and the rendering process. The lighting estimator receives the segmented image and outputs the lighting settings that describe the real environment lighting. The game engine receives the lighting settings and adjusts the virtual environment to match the real environment. The hands' montage process overlay the segmented image in the virtual world image to produce the mixed reality scenario.

4.3.1 Input Processing

The input processing step of the hand-based mixed reality framework consists of taking the input picture and outputs a suitable image for lighting estimation and mixed reality rendering. The main steps that constitute the input processing are:

Human hands detection. This step analyzes the input picture providing information if the user's hands are visible in the picture. It dictates if a captured frame should proceed to the lighting estimation model or should be discarded.

Hands segmentation. This processing step removes all pixels of the input picture that does not belong to the user's hands. The output of this process is an image with the user's hands segmented from the image.

Ideally, every frame of the camera feed should be processed; hence, both the hands' detection and hands segmentation processes should be computationally efficient and suitable for real-time applications.

The implementation of this framework in this thesis considered that the lighting estimation models are robust enough to estimates lighting even when other parts of the user's upper limbs are present (such as the arms and upper-arms) — leading to the conclusion that a skin segmentation algorithm should be sufficient to capture the desired portion of the input picture.

A threshold-based skin segmentation algorithm performed the removal of the pixels that do not belong to the user's upper limbs. The algorithm makes use of images under different color spaces to identify the pixels in the image that contain a color intensity that characterizes the human skin. The color spaces used in this algorithm are the RGB, HSV, and YCbCr color space [115]. The threshold values are based on empirical test results presented in the work of Kolkur et al. [56].

The resulting image of the threshold-based skin segmentation is a binary mask that marks pixels that belong to the skin. The mask has some noisy regions around the edges, and some small regions that belong to the skin were not marked correctly by the segmentation algorithm.

Additional steps were implemented to improve the generated mask. A closing morphological image processing operation was performed on the mask to fill small holes in the segmented skin regions of the image. A normalized box filtered blur effect was applied to the mask to smooth the edges of the segmented area. The next step was to find the contours of the segmented area and flood fill the interior of the segmented area to obtain the binary mask.

A bitwise AND operation is applied to the input image and the binary mask pixels to produce the segmented image containing only the skin pixels. The hand detection process was performed by taking advantage of the segmentation mask to detect if a significant portion of the hand is visible in the input picture. The sum of the pixel values of the mask image would give the portion of pixels that belongs to human skin in the input picture. A simple threshold value, based on the sum of pixels and the total area of the input picture, was utilized to output a boolean value. In this work, we use a heuristic: if more than 10 percent of the pixels in the segmented image belongs to human skin, then it is considered that the human hand is visible in the frame.

4.3.2 Lighting Estimation

The lighting estimation process in the mixed reality framework is the process that receives as input the image of the hands with the background removed and outputs a suitable representation of the environment lighting.

Three lighting estimation models were developed in this thesis. Two of then assumed the hand-based light probe hypothesis, and thus make use of the proposed hand-based mixed reality framework. The main difference between the developed hand-based lighting estimation methods is the environment lighting representation utilized.

The PLS estimator utilizes point light-source as the representation of the dominant light source of the environment lighting. Chapter 5 presents this model and the implications of this light-source representation in the mixed reality environment.

The SHLP estimator utilizes an area light-source encoded in a spherical harmonics light probe to represent the environment lighting of the scene. Chapter 6 presents the model and detail it's implementation.

The ELP estimator is a third lighting estimation model developed in this thesis. The ELP estimator does not assume the hand-based light probe hypothesis and aims to estimate a light-probe image of the environment for a generic use-case. The ELP estimator model is discussed separately in Chapter 7 of this thesis.

The output of the lighting estimation process is feed to the lighting adjustment step of the mixed reality framework. The lighting adjustment is implemented directly in the mixed reality application.

4.3.3 Lighting Adjustment

The lighting adjustment is employed by the mixed reality application. It is responsible for receives the environment lighting estimated in the previous step and adapt the virtual world lighting settings.

The implementation of this step in this thesis methods consists of directly adjusting the virtual world lighting by one of the following approaches, depending on the lighting estimation model utilized:

Directly moving the primary point light source to the estimated lighting position of the PLS model; Changing the ambient light component of the environment lighting with the estimated SH coefficients of the SHLPE model; Changing the environment light probe-image of the ELP model.

These strategies of lighting adjustment are sufficient for the experiments and validation tests of the lighting estimation models employed in this thesis. Although more elaborated approaches make an exciting subject of future researches that could potentially augment the user's experiences concerning immersion and presence in the mixed reality applications.

4.3.4 Hands Montage

The hand's montage is an optional step in the hand-based mixed reality framework. It consists of overlaying the virtual world image with the segmented image of the user's hands. This image compositing process can be accomplished by a two-pass rendering strategy, first rendering the entire virtual world and then a second pass directly rendering the segmented image of the user's hands as an alpha-blended texture in the screen-space [91].

The hand's montage is relevant to the implementation of hand-based augmented virtuality using virtual reality HMDs. However, it is unnecessary for augmented and mixed reality applications using light field displays [39], in which virtual objects are directly projected in the real-world environment.

4.4 Deep Learning Approach to Lighting Estimation

The lighting estimation task proposes a solution to an ill-posed problem. An ill-posed problem is a mathematical term defined by Jacques Hadamard [35], describing a problem that does **not** satisfy some properties concerning the solution of this problem: The solution's behavior changes continuously with the initial conditions, the solution to the problem exists, and the solution is unique; the latter property is not satisfied by the solutions for the problems tackled in this thesis.

This characteristic makes the development of lighting estimation algorithms a challenging task. Several works (See Chapter 3) successfully make use of deep learning techniques to generate solutions to ill-posed problems by applying learning by data approaches instead of hand-crafted algorithms. In this work, we follow this approach by employing convolutional neural networks to estimates the environment lighting.

The convolutional neural networks require a dataset with a high number of examples to learn a robust and generalist representation of the problem to be solved. Depending on the task to be solved, a dataset with sufficient examples could not exist and can be challenging to obtain. The lighting estimation is one such task. There is not an accessible way to obtain a high number of examples containing the measured environment lighting of real-world pictures.

An alternative approach is to generate synthetic examples through computer graphics, rendering virtual environments with known lighting settings. Methods that use synthetic datasets assumes that the estimation model is robust enough to provide results that generalize well to real-world images.

The first two methods developed in this thesis, The PLS, and the SHLP estimators, utilizes a synthetic dataset to estimate the lighting settings of the environment. The Section 5.2 and Section 6.2 describe the synthesis of the dataset for both methods.

Chapters 5, 6 and 7 present the implementation details of the convolutional neural networks architecture, dataset, and training process for the respective lighting models developed in this thesis.

Chapter 5

Point-Light Source Estimator

The point-light source (PLS) estimator is a lighting estimation model that uses a point light source representing the dominant light of the environment lighting. The focus of the PLS estimator is to estimate the position of the light source in the scene.

The PLS estimator describes the lighting estimation as a multinomial classification task. The estimator consists of a convolutional neural network that classifies an input instance as one of the classes provided in the dataset. Each class represents a light-source position that defines the characteristics of the environment lighting. The PLS estimator characterizes the environment lighting by grouping examples into a discrete number of possible classes.

One implication of representing the lighting as a classification task is that the space of possible lighting settings is discretized based on the number of classes. A low number of classes can result in under-representation of the real environment lighting, producing low-accuracy estimations. While an excessive number of lighting settings creates a group of classes in which visually similar images are grouped in different classes, difficulting the learning process. Experiments were performed with four levels of discretization and are discussed later in this chapter.

The PLS estimator assumes the hand-based light probe hypothesis (Sec 4.2); thus, it requires an input picture of the environment with the user's hands visible. The input picture is captured from the user's perspective by a camera positioned in HMD. Hence, the input picture is referred to as an egocentric view of the environment.

The output of the PLS estimator is an integer number in the range of [0, n], where n is the number of classes. The integer identifies which class characterizes the environment lighting of the input image, consequently giving the light-source position of the dominant

light in the environment.

The PLS estimator is based on convolutional neural networks and thus requires a suitable dataset. The PLS dataset was created to support the training of the PLS estimator; the dataset is populated by examples representing the lighting setting classes. The dataset synthesis is discussed in the next section of this chapter.

The main characteristics of the PLS estimator are summarized as follows:

- It utilizes a point-light source to represent the environment lighting;
- It is posed as a multinomial classification task;
- It estimates the spatial position of the dominant light source of the scene based on the classification into one of the determined classes of lighting settings;
- It makes uses of the hand-based light probe hypothesis (Sec 4.2);
- The input data is a picture of the environment taking in an egocentric view.
- The output data is an integer number that identifies one of the lighting settings classes. A lighting setting class contains information about the three-component vector representing the 3D position of the environment light source.
- The estimator utilizes a synthetic dataset to learn the environment lighting by examples.

5.1 Input Data

The input data of the PLS estimator is a picture of the environment scene. In a mixed reality context, the picture is taken from the user's perspective by a camera located in the HMD. The consumer version of the HMDs currently available in the market already have built-in cameras (Oculus Rift S, Oculus Quest, HTC Vive, Valve Index, and the vast majority of windows mixed reality HMDs). Hence, the applicability of this method is straightforward concerning the current generation of HMDs, not requiring any additional hardware to capture the input image.

It is assumed that the input image is similar to those obtained by one of the cameras located in the HMD; thus, the PLS estimator utilizes a dataset with images that mimics those camera specifications. In particular, we choose the same parameters as the HTC Vive built-in camera. The specifications include the camera field-of-view and image resolution.

Since the PLS estimator is based on the hand-based light probe hypothesis, it is required to perform the preprocessing step described in Section 4.2 of the previous chapter. The output image of this preprocessing step is a segmented image of the user's hands to be used for the lighting estimation.

It is worth note that preliminary experiments of lighting estimation in the raw input image were performed. The approach of lighting estimations on images without the segmentation preprocessing produced poor estimation results. Hence, the aforementioned approach was discontinued.

The reasons for the poor performance of the raw images are associated with the convolutional neural network ignoring the usage of the hands as light-probe and trying to infer the lighting settings from other lighting cues in the pictures. Since the provided dataset contained only a limited number of environments, it was not possible to train a reliable model using the raw images. The proposed preprocessing step counters this problem by removing the background environment, thus, forcing the neural network to learns from the lighting cues presents on the hands light-probe.

5.2 Point Light Source Dataset

At the time of development of the PLS estimator, there was no publically available dataset suitable for the environment lighting estimation using point light-source models. For this reason, a complete synthetic dataset was created. The PLS dataset contains examples of environments with varying lighting conditions. The examples were composed of images containing human hands under known lighting settings. A single point light-source was placed in a virtual 3D scene in order to generate the lighting setting.

3D human hands models were authored to represent the user in the mixed reality application. Two geometry meshes were created accounting to represent both male and female characteristics. Additionally, three different skin materials were created, accounting for racial variations. The hands model is rendered using a screen-space skin shader model that approximates the diffuse subsurface scattering of human skin [49].

The 3D virtual hands were animated to simulate the most expected interactions performed by the user in mixed reality games and simulations. The animations introduce



Figure 5.1: The PLS dataset scene setup: The virtual scene is composed of a virtual camera, a 3D hand model, and a single point light-source model.

variation in the dataset by providing a set of images under different hand poses. This variation is a desirable characteristic since the lighting estimation model should be equivariant in terms of the user's hands pose. The animations were created to simulate the actions of walking, jumping, pushing an object, grabbing an object, and punching an object.

The positioning of the virtual hands is of significant importance due to the goal of simulating the input data of the real-world picture taking from the HMD camera. The virtual 3D hand model is positioned right in front of the virtual camera to simulate the user's placement in an egocentric (first-person) view. An individual point light-source is present in the scene. Figure 5.1 illustrates the basic scene setup. The point light is placed with a fixed distance of 200 cm of the camera position.

To represent the lighting configurations, we use an algorithm to create an approximately even distribution of possible light position in the scene. The Fibonacci lattice distribution algorithm [78] distributes n points on the surface of a sphere. We move the point light-source position to one of these points and render the scene to generate an example in our dataset. The pseudocode of the Fibonacci lattice distribution algorithm is listed in the Algorithm 3.

Algorithm 3 Lattice distribution algorithm	
1: procedure DISTRIBUTION (n)	\triangleright n is the number of samples
2: $offset \leftarrow 2.0/n$	
3: $increment \leftarrow (3.0 - \sqrt{5.0}) * \pi$	
4: pointList.create()	\triangleright Create an empty list
5: for $i = 0 \rightarrow n$ do	
6: $y \leftarrow ((\underline{i * off set}) - 1.0) + (off set/2.0)$	
7: $r \leftarrow \sqrt{1-y^2}$	
8: $phi \leftarrow ((i+1) \mod n) * increment$	
9: $x = \cos(phi) * r$	
10: $z = \sin(phi) * r$	
11: $pointList.add(x, y, z)$	
12: end for	
13: return <i>pointList</i>	
14: end procedure	

Each example in the dataset is composed of the image containing the rendered scene and the position of the point light-source present in the virtual scene during the rendering process.

We created four variations on the PLS dataset, those variations account for the number of possible light source positions in the space. Thus, specifying the level of discretization utilized for the lighting positioning. The discretization of the lighting position was 5, 25, 50, and 100 possible lighting positions on the sphere surface.

The distributions of point light-source on the spherical surface are shown in Figure 5.2. With the exception of the five-light distribution, the point-light distribution was performed by the Fibonacci lattice distribution algorithm. The five-light distribution was created by manually positioning the light source into five predefined positions relative to the camera position; those positions were front-view, left-view, right-view, top-view, and back-view positions.

Due to the animated nature of our hands model, the hands model could fall off the virtual camera viewport, resulting in an entirely black image. Because of that, we discard any rendered frame where the hands are not visible by the virtual camera. Samples of the PLS dataset are shown in the Figure 5.3. The number of examples generated to compose the 4 variations of the PLS dataset is shown in the Table 5.1. We separate the examples in disjoint-sets of training, validation and test sets using, respectively, the ratio of approximately 65%/25%/10% of the total examples.



Figure 5.2: Distribution of points on the surface of a sphere: Points are distributed on the surface of a virtual sphere. The point is a light position candidate for the PLS estimator. Top left: five points candidates. The five points were chosen manually to represent the front, left, right, top, and back directions. Top right: twenty-five points candidates from the Fibonacci distribution algorithm. Bottom left: fifty points candidates from the Fibonacci distribution algorithm. Bottom right: one hundred points candidates from the Fibonacci distribution algorithm.

Lighting	Total	Training	Validation	Test
$\mathbf{settings}$	examples	\mathbf{set}	\mathbf{set}	\mathbf{set}
5	4527	2943	1133	451
25	20761	13496	5194	2071
50	41044	26676	10269	4099
100	83799	54471	20965	8363

Table 5.1: Number of examples generated for each variation of the PLS dataset.



Figure 5.3: The PLS Dataset sample images: Resulting sample images of the PLS dataset synthesis method. Different skin materials and mesh geometries were used to create variety in the dataset. A label describes the 3D position of the dominant point light-source associated with each class in the PLS dataset.

5.3 Point Light Source Estimator Architecture

The PLS estimator makes uses of a convolutional neural network based on the Residual network architecture; this network architecture was discussed in Section 2.4.3. This architecture was chosen due to its high accuracy in image classification tasks (The ResNet implementation of the residual network architecture was the winning model of the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) of 2015 [97]). At the time of development of the PLS estimator, the ResNet was the most performant neural network of the state-of-the-art for image classification regarding classification accuracy.

The architecture of the PLS estimator convolutional network is shown in Table 5.2. The network adopted has 49 convolution layers + 1 fully connected layer resulting in a 50 layers network. The first layer of the network is a (7×7) convolution layer with the stride of 2 pixels, and a zero-padding of 3 pixels, outputting 64 filters. This layer is followed by a max-pooling layer with a stride of 2 pixels. This first convolution layer utilizes a bigger kernel size $(7 \times 7 \text{ pixels})$ in an attempt to embed the input image in a set of rich primary features.

The usage of larger kernel size yields a set of generic features spread across the image, while the usage of smaller kernel sizes extracts localized features of the image. A disadvantage of larger kernel sizes is the increased computational complexity. The computational cost of the convolution layer scales with the number of feature maps in the layer's input. Since the input image has a relatively small number of feature maps (3 feature maps, one for each color channel) when compared to the feature maps in the hidden layers, it is desirable to use larger kernel size only at the beginning of the network. All the other convolution layers in this architecture utilize convolution layers with 3 x 3 kernel size.

The next layers of the network are created by stacking the residual network building blocks (described in the Caption 2).

The initial building block has a (3×3) convolution layer with 64 filters, the last layer of this building block outputs a feature map of size 256. We serially stack this block three times (as shown in the repeat column of Table 5.2). Following the same building scheme, the next building block use (3x3) convolution layers with 128 filters and outputs a feature map of size 512. The same scheme is repeated two more times by convolution layers of 256 and 512 filters, and outputs of 1024 and 2048 feature maps, respectively. The last portion of the network is composed of a (7×7) average pooling layer followed by a fully-connected layer with a ReLu activation function. A softmax loss function is utilized

Kernel Size	Stride	Zero-padding	Output	Repeat
$7 \ge 7$ Convolution	2	3	64	1
3 x 3 Max Pooling	2	0	04	1
$1 \ge 1$ Convolution	1	0	64	
$3 \ge 3$ Convolution	1	1	64	3
$1 \ge 1$ Convolution	1	0	256	
$1 \ge 1$ Convolution	1	0	128	
$3 \ge 3$ Convolution	1	1	128	4
$1 \ge 1$ Convolution	1	0	512	
$1 \ge 1$ Convolution	1	0	256	
$3 \ge 3$ Convolution	1	1	256	6
$1 \ge 1$ Convolution	1	0	1024	
$1 \ge 1$ Convolution	1	0	512	
$3 \ge 3$ Convolution	1	1	512	3
$1 \ge 1$ Convolution	1	0	2048	
7 x 7 Avg. Pooling	1	0	2048	1
Fully Connected	-	-	5/25/50/100	1

Table 5.2: The Point Light Source Estimator Network Architecture

to yield a probability distribution that describes the probabilities of the input image to belongs to one of the lighting setting classes.

5.4 The Residual Network Training

The training process of the residual network was implemented using the Caffe Framework [47]. The 50-layers residual network receives as input images with a size of 455 pixels width and 256 pixels height. Different from the usual input size for residual networks that are images with 224 x 224 pixels.

The usage of rectangular input size is the outcome of the methodology utilized to generate the PLS dataset, that mimics the HTC Vive HMD camera. Therefore, the dataset contains images with a 16:9 aspect-ratio and a spatial resolution of 1280 x 720 pixels. An image scale operation (bicubic interpolation) was performed in the input image to provides the network with the desired input size of 455 x 256 pixels.

Preliminary experiments were performed by training the network with square crops of the input image. The resulting lighting estimation was not satisfactory, the reason behind the difficulty in estimating the lighting settings in the image crops is that lighting estimation is not equivariant to translation, thus cropping an area of the original image resulted in poor representation of the environment and wrong classifications. The residual network is trained for 80 epochs, with a learning rate of 0.0001. The training batch size was defined as 32 examples; this number was determined by the available GPU memory in our training system. The standard Stochastic Gradient Descent (SGD) [7] algorithm was used for the minimization of the loss function.

The loss function used was the Softmax function. This loss function is appropriate for the classification task since it outputs a probability density function of the classes in the dataset. We train one network for each variation in the PLS dataset, totaling four networks. One for each level of discretization in the lighting settings. The only difference in the network architecture between the different lighting configurations is the output of the last layer, which corresponds to the number of classes: five, twenty-five, fifty, and one hundred classes.

5.5 Experiments and Results

The training process was executed on a machine with the following specification:

CPU Intel i7 4790 @ 3.6Ghz

Memory 24 GB of DDR3 RAM

GPU Nvidia Geforce Titan X with 12GB of GDDR5.

OS Ubuntu 16.04

Tests on the inference process were executed on a machine with the following configuration:

CPU Intel i7 3770k @ 3.9Ghz

Memory 8GB of DDR3 RAM

GPU Nvidia GeForce GTX 680 with 2GB of GDDR5.

OS Windows 10

5.5.1 Training Time and Accuracy

The training time of the residual network in the PLS estimator increases significantly as the number of light settings increases, as shown in Table 5.3. It is important to note that the training phase is a process that must be executed only once. In the runtime, only the inference phase is executed. The inference time in the CPU took 0.53 seconds. The inference time in the GPU took 16ms, making it suitable for real-time environments. The number of parameters learned does not increase significantly because only the last layers, the fully connected layer, is changed in the architecture.

Table 5.3: Training statistics and top-1 accuracy for the point light source estimator on the test dataset.

Lighting	Top-1	Training	Learned
$\mathbf{settings}$	accuracy	Time (h)	Parameters
5	93.87~%	00:19	23,541,231
25	83.15~%	03:42	$24,\!585,\!241$
50	82.73~%	05:01	$25,\!609,\!266$
100	81.51~%	10:16	$27,\!657,\!316$

The output of the softmax layer in the residual network classifier is a discrete probability distribution encoded in an array of size n, where n is the number of classes in the classification problem. The top-1 accuracy takes only the class with the most probability as a result of the classification. The top-1 accuracy of the trained networks, as shown in Table 5.3, decreases as the number of possible classes increases. This behavior can be explained by the residual network classifying classes that are spatially close but do not match the correct test label in the test process of the CNN training.

5.5.2 Synthetic Scenario

An experiment was executed to check the accuracy of the lighting estimation for a synthetic image with a previously unseen lighting setting. Figure 5.4 shows the resulting image of the PLS estimator in this experiment. The figure shows the lighting estimation for a scene containing the virtual hands and two objects with simple geometry: a cube and a sphere.

A randomly selected pose of the virtual hands was selected to generate each test example. The hands were placed in the virtual environment in the selected pose in combination with a point light source under a randomly generated light position. The rendered virtual scene is the ground-truth image of the experiment.

The image containing only the virtual hands (eliminating the cube and sphere of the image) is fed to the PLS estimator, this process is utilized to simulate the image segmentation pre-processing step from the regular real-world estimation process. The



Figure 5.4: PLS lighting estimation on synthetic input image. Left column: ground-truth image generated from the 3D hands and two 3D objects (3D cube and 3D sphere) illuminated by a known point light source. Center column: scene illuminated by the point light source estimation. Right Column: The difference image of the ground-truth image and the estimated result image.

Row in	Root mean-square	Normalized
Figure 5.4	error	cross-correlation
1	580.323	0.86520
2	349.108	0.86584
3	921.224	0.86216
4	574.501	0.86499
5	888.459	0.86379
6	1891.270	0.85247

Table 5.4: lighting estimation on the synthetic test. RMSE and NCC statistics for Figure 5.4 examples.

output class of the PLS estimator is used to generate the resulting image of the predicted light source position.

The difference image is employed to illustrate the error in the estimation process (right column in figure 5.4). The difference image is created by pixel-wise subtracting intensity values of the pixels in the resulting image from the intensity values from the pixels in the ground truth image; the resulting difference image is converted to grayscale for better visualization.

The estimations of the one-hundred lighting settings residual network were used to generate the images in Figure 5.4. The use of a random point light position implies that the values chosen for the point light position do not match the predefined positions in the one hundred possible light positions of the PLS dataset. The estimator was capable of outputting a classification that generates images with a plausible lighting configuration. The first two rows of images in the Figure 5.4 show lighting estimations that are close to the ground truth (RMSE error: 580.323 and 349.108, respectively), while the third and the last two rows of images show the results of the experiment with a higher associated error (RMSE error: 921.224, 888.459 and 1891.27, respectively). Even in the cases that the estimation differs from the ground truth, the general appearance of both images are similar and present a plausible lighting and shading. The root mean square error (RMSE) and the normalized cross-correlation (NCC) for all the images in the Figure 5.4 are shown in the Table 5.4.

5.5.3 Augmented Virtuality Scenario

An experiment was made to demonstrate the visual impact of the correct dominant light position in an augmented virtuality environment. Figure 5.5 shows the resulting images of the experiment.



Figure 5.5: Lighting adjustment on Augmented Virtuality application. Left: the input image containing synthetic human hands, the background was removed from the original image to simulate the skin segmentation algorithm behavior in the nonsynthetic image input scenario. Center: The input image is overlayed into the virtual world; the lighting settings in the virtual scene remain unchanged. Right: the lighting settings in the virtual scene is adjusted to match the recognized illumination in the synthetic hands. Figure from [77].

We provide an input image containing the virtual hand. All the background pixels were removed during the rendering. A montage was created by overlaying the input image and the virtual environment. The dominant light source is represented by a point light source located around the user's position in the virtual environment. The input image is feed to the PLS estimator, and the resulting lighting condition is used to adjusting the virtual environment's dominant light. We choose not to change the light sources that were originally present in the virtual scene. The resulting montage with the dominant light source matching the synthetic hand lighting is shown in the right column of Figure 5.5.

The visual perception of the correct light position in the image dramatically improves the immersion by correctly blending the hands and the virtual environment. In those images, only the light position was changed. Adjustments such as light color, exposition, and light intensity can improve the experience and are exploited in the SHLP estimator described in Chapter 6.

5.5.4 Level of discretization

An experiment was conducted to compare the impact of increasing the level of discretization (number of lighting settings classes) in the lighting estimation model. Figure 5.6 shows images comparing the different levels of discretization in the number of light settings for the classification task. The scene was rendered with a virtual scenario and a real human hand in an application of Augmented Virtuality. The base image (top image in Figure 5.6) is generated with the one-hundred lighting settings PLS estimator. Selected areas of the base image were selected and cropped to highlight the differences between the lighting in the base image and the images generated with the remaining PLS estimators (fifty, twenty-five, and five lighting settings).

The increment in the number of lighting settings produces a drastic difference in the case of five lighting settings versus one hundred lighting settings. This indicates that the five lighting settings are not sufficient to produce an accurate estimation of the environment lighting. Increasing the number of lighting settings to 25 and 50 lighting settings dramatically improves the estimation by allowing the dominant point light source to be placed in more plausible positions in the 3D space.

The difference images in Figure 5.6 indicates that the resulting lighting in the twentyfive and fifty lighting settings do not highly differ from the one-hundred lighting settings. This result implies that a twenty-five lighting setting is sufficient for generating plausible



Figure 5.6: Point Light Source estimation with different lighting settings discretizations. Top: An image of a real human hand inserted in the virtual environment, the lighting settings of the virtual environment were adjusted to match the estimated lighting condition of a CNN trained with 100 lighting settings. Crops of the image compare the visual appearance of the lighting estimation by a CNN trained with, respectively, 100, 50, 25, and 5 lighting settings. Each column shows the color image and the difference image with the 100 lightings settings image. Figure from [77].

and consistent lighting in the mixed reality environment.

An additional advantage of twenty-five lighting-settings corresponds to the time necessary in the training phase of the residual network. The training of the twenty-five lighting-settings CNN was considerably faster than for a hundred lighting settings CNN (as shown in Table 5.3). This occurs due to the number of examples in the utilized dataset.

Chapter 6

Spherical Harmonics Light-Probe Estimator

The Spherical Harmonics Light-Probe (SHLP) estimator is a lighting estimation model that uses a spherical harmonics lighting to represent the environment lighting. The focus of the SHLP estimator is to estimate the coefficients of a spherical harmonics basis functions that constitute an area-light model of the environment lighting. The SHLP estimator describes the lighting estimation as a regression task. The estimator consists of a convolutional neural network that regresses nine float numbers representing the spherical harmonics coefficients.

Using a spherical harmonics basis to represent the environment lighting is an approximation of using a light-probe image. This approximation implies that a low number of coefficients could lead to a substantial loss of information, while a high number of coefficients leads to a better approximation but reduces the advantages of this representation, such as the compact format.

Similar to the PLS estimator presented in the previous chapter, the SHLP estimator also assumes the hand-based light probe hypothesis (Sec 4.2), requiring an input picture of the environment with the user's hands visible. The input picture is captured from the user's perspective by a camera positioned in HMD. The output of the SHLP estimator is a nine-component vector, in which each component is a float number in the range of [0, 1]. The float numbers are coefficients of a spherical harmonics basis that characterizes the environment lighting of the input image.

Since the SHLP estimator is based on convolutional neural networks, it requires a suitable dataset of the input image and the sh encoded environment lighting. The SHLP dataset was created to provides support for the training of the SHLP estimator. The dataset synthesis is discussed in the next section of this chapter.

The main characteristics of the SHLP estimator are summarized as follows:

- It utilizes spherical harmonics lighting to represent the environment lighting;
- It is posed as a regression task;
- It estimates nine coefficients of a spherical harmonic basis function;
- It makes uses of the hand-based light probe hypothesis (Sec 4.2);
- The input data is a picture of the environment taking in an egocentric view.
- The output data is a nine-component vector. Each component of this vector is a float number that determines the coefficient value for the spherical harmonics lighting.
- The estimator utilizes a synthetic dataset to learn the environment lighting by examples.

6.1 Input Data

The SHLP estimator assumes the hand-based light probe hypothesis. Thus the input data of the SHLP estimator is a picture of the environment scene with the user's hands visible. The input image of the SHLP is similar to the PLS estimator presented in the previous chapter.

A summary of the properties of the input picture is listed below:

- An egocentric view of the real-world environment captured by the HMD camera.
- The picture resolution and field-of-view mimic the HTC Vive built-in camera.
- The user's hands should be visible in the picture.
- A preprocessing step should be performed to remove the background pixels of the image, effectively segmenting the hands.

6.2 Spherical Harmonics Light Probe Dataset

The training process of the convolutional neural network that estimates the spherical harmonic light probes requires a dataset containing the input images and the respective values for the SH light probe coefficients. For this reason, similar to the PLSE, a synthetic dataset (The SHLP Dataset) was created. An example of the SHLP dataset consists of an image containing human hands under different lighting conditions. The lighting conditions are represented by a spherical harmonics encoded environment light probe.

The SHLP estimator treats the lighting estimation process as a regression task. Similar to the classification task, the regression task estimates values for a set of outputs. However, the regression task estimates a continuous value for one or more variables, while the classification task estimates a probability function that gives the probability of input to belong to the label classes in the training dataset.

The regression convolutional neural network is the core of the SHLP estimator. The neural network outputs a set of spherical harmonics coefficients that represent a light probe in the real environment. Each coefficient has a value in the range [-1, 1].

Each entry in the SHLP Dataset is composed of the image containing the rendered scene and the nine SH coefficients used during the rendering process. We use a secondorder spherical harmonics basis to approximate the environment lighting. A light probe composed by a set of nine SH coefficients is sufficient to approximate the low frequency, diffuse, lighting condition of a real light probe [92]. This representation allows multiple light sources with distinct intensities in the resulting lighting probe.

To create representations of various lighting conditions, we generate two SH light probes formed of nine coefficients each. Then we combine all coefficient values in the light probes to create 2^9 potential lighting configurations. The coefficients are generated by randomly sampling real numbers in the range [-1, 1].

To synthesize the SH lighting dataset, we use the same 3D hands model as described in Section 5.2 of the previous chapter. We change the lighting settings in the virtual environment using the SH coefficients in the rendering pipeline [29]. We use the Unreal Engine 4 for 3D rendering the scene.

We discard any rendered frame in which the hands are not visible by the virtual camera. Samples of the SHLP dataset are shown in Figure 6.2. We generate 368946 examples to compose the spherical harmonics light probe dataset.



Figure 6.1: The SH Light Probe Dataset scene setup: The scene is composed of a virtual camera, a 3D hand model, and a set of nine SH coefficients that represents the lighting of the scene.



Figure 6.2: The SHLP Dataset sample images: Resulting images of the dataset synthesis process. Different skin colors and mesh geometry were used to create the dataset. There is a label describing the nine SH coefficients lighting settings associated with each sample in the dataset.

6.3 Spherical Harmonics Light Probe Estimator architecture

The SHLP estimator makes uses of a convolutional neural network based on the Dense Convolution Network architecture [45]. This network architecture was discussed in Section 2.4.4. This architecture has a classification accuracy comparable to the residual network employed in the PLS estimator of the previous chapter; however, the dense convolution network has substantially fewer parameters to learn. The 50 layers ResNet have approximately 25 million learning parameters, while the 121 layers DenseNet network has approximately 8 million parameters with slightly better accuracy for the ImageNet classification task. Hence, we elect to use the DenseNet architecture for the SHLP estimator, taking advantage of the reduced number of parameters to speed up the neural network training time and allowing a fast inference when compared to the ResNet architecture. At the time of development of the SHLP estimator, the DenseNet was the most performant method in the state-of-art for image classification and object detection, regarding accuracy and number of learned parameters.

The SHLP estimator tackles the lighting estimation problem as a regression task. For this purpose, the loss function used was the root mean squared error (RMSE). The output of the network is a nine components vector containing the spherical harmonics coefficients of the environment lighting.

The network adopted has 120 convolution layers + 1 fully connected layer resulting in a 121 layers network. The network architecture uses the standard growth rate of k = 32. The architecture of the SHLP estimator convolutional network is shown in Table 6.1.

The first layer of the network is a (7×7) convolution layer with the stride of 2 pixels, and a zero-padding of 3 pixels, outputting 64 filters. This layer is followed by a (3×3) max-pooling layer with a stride of 2 pixels. The next layers of the network are composed of stacking dense blocks. A transition layer is utilized to connect two consecutive dense blocks. The transition layer is composed of a 1x1 convolution layer (responsible for halving the number of feature maps) and a 2x2 average pooling layer (responsible for halving the feature map size).

The first dense block has six (3×3) convolution layers; the last convolution layer outputs 256 feature maps of size 56. The transition layer halves the feature map size and the number of feature maps of the previous layer resulting in 128 feature maps with a spatial size of 28. The architecture follows the same scheme os stacking dense blocks and using the transition layer between two consecutive dense blocks; the final architecture utilizes a stack of 4 dense blocks. The number of (3×3) convolution layers in each dense block is shown in the repeat column of Table 6.1. The last layer of the network is a fully connected layer activated by a hyperbolic tangent function; this activation function was chosen to limit the output of the fully-connected layer to values in the range of [-1, 1].

Kernel Size	Stride	Zero-padding	Feature Maps	Repeat	
7 x 7 Convolution	2	3	64	1	
$3 \ge 3$ Max Pooling	2	0	04	1	
1 x 1 Convolution	1	0	256	6	
$3 \ge 3$ Convolution	1	1	$(64+32\times 6)$	0	
1 x 1 Convolution	1	0	198	1	
2 x 2 Average Pooling	2	1	120	1	
1 v 1 Convolution	1	0	519		
2 x 2 Convolution	1	1	$(129 \pm 22 \times 12)$	12	
5 X 5 CONVOLUTION	1	0	$(120 + 32 \times 12)$		
1 x 1 Convolution	1	0	256	1	
2 x 2 Average Pooling	2	1	200	1	
1 x 1 Convolution	1	0	1024	94	
$3 \ge 3$ Convolution	1	1	$(256 + 32 \times 24)$	Z4	
1 x 1 Convolution	1	0	519	1	
2 x 2 Average Pooling	2	1	312	1	
1 x 1 Convolution	1	0	1024	16	
$3 \ge 3$ Convolution	1	1	$(512 + 32 \times 16)$	10	
7 x 7 Avg. Pooling	1	0	1024	1	
Fully Connected	-	-	9	1	

Table 6.1: The Spherical Harmonics Light Probe Estimator Network Architecture.

6.4 Model Training

The network architecture was implemented in the Tensorflow library.

The 121 layers dense convolutional neural network receives as input images with a spatial resolution of 512 x 256 pixels. The network was trained for 60 epochs with a learning rate of 10^{-4} . The training batch size was defined as 7 examples per mini-batch; this number was determined by the available GPU memory in our training system.

The spherical harmonics light probe dataset was split into training, validation, and test sets. The splitting ratio utilized was approximately 70% of the data for training (258262 samples), 20% for validation (73789 samples), and the remaining 10% for testing (36895 samples). We ensure for our split sets that a sample generated from a lighting

setting appears only in a single set; thus, the split produces disjoint sets.

The Adam optimizer was used in the first 20 epochs of the training process. The Stochastic Gradient Descent (SGD) [7] algorithm was used in the remaining 40 epochs of the training. This strategy was adopted to accelerate the optimization convergence at the beginning of the training while improving the generalization of the model by using the SGD algorithm.

6.5 Experiments and Results

The training process was executed on a machine with the following specification:

CPU Intel i7 4790 @ 3.6Ghz

Memory 24 GB of DDR3 RAM

GPU Nvidia Geforce Titan X with 12GB of GDDR5.

OS Ubuntu 16.04

Tests on the inference process were executed on a machine with the following configuration:

 ${\bf CPU}$ Intel i
7 $3770{\bf k}$ @ 3.9Ghz

Memory 8GB of DDR3 RAM

GPU Nvidia GeForce GTX 680 with 2GB of GDDR5.

OS Windows 10

6.5.1 Training Time and Accuracy

The training time of the dense convolution network for the SHLP estimator took 40 hours. It is important to note that the training phase is a process that must be executed only once. In the run-time, only the inference phase is executed. The inference time in the CPU took 0.69 seconds, while in the GPU, the process took 18 milliseconds. The dense convolution network learned approximately 8 million parameters. The RMS error of the trained networks was 0.0573 against the test images.



Figure 6.3: SHLP estimator on synthetic input. Left Column: Ground truth image generated from the 3D hands and two 3D objects (3D cube and 3D sphere) illuminated by a known spherical harmonics coefficients. Center Column: Scene illuminated by the SHLPE. Right Column: The difference image of the ground truth and the estimated result.

6.5.2 Synthetic Scenario

Figure 6.3 shows the resulting image of the SHLPE lighting estimation in a scene containing the 3D human hands and two objects. Randomly selected poses of the 3D hands were selected and placed in the virtual environment to generate each example. A randomly generated spherical harmonics light probe to render the scene resulting in the ground truth images. The image containing only the 3D human hands are fed to the SHLP estimator, the output coefficients are used to generate the resulting image of the predicted light probe. A difference image is used to illustrate the error in the estimation process. The difference image (right column in Figure 6.3) is created by subtracting pixel's intensity in the resulting image from the pixel's intensity in the ground truth image; the resulting difference image is converted to grayscale for better visualization.

The SHLPE was adopted to generate the images in Figure 6.3. The use of a random light probe generated from random sampled spherical harmonics coefficients implies that the values chosen for the light probe do not match the predefined light probe in the SHLP dataset. The estimator was capable of outputting spherical harmonics coefficients values that generate an image with a plausible lighting configuration. The first row of images in

Table 6.2 :	Lighting	estimation	on the	syntethic test	RMSE	and NCC	statistics	for F	Figure
6.3 examp	oles.								

Row in	Root mean-square	Normalized
Figure 6.3	error	cross-correlation
1	495.127	0.865148
2	975.335	0.862725
3	1367.840	0.859666

the Figure 6.3 shows lighting estimation that is close to the ground truth (RMSE error: 495.127), while the second and the third rows of images show the results of the experiment with a higher associated error (RMSE error: 975.335, 1367.84, respectively). Even in the cases that the estimation differs from the ground truth, the general appearance of both images is similar and present plausible lighting. The Root Mean Square Error (RMSE) and the Normalized Cross-Correlation (NCC) for all the images in Figure 6.3 are shown in Table 6.2.



Figure 6.4: SHLPE and PLSE [77] lighting estimation on complex lighting scenarios.

6.5.3 Spherical Harmonics vs Point Light Estimators

The estimated prediction of the SHLP estimator was compared to the estimated prediction of the PLS estimator. Increasing the complexity of the lighting in the scene, by adding additional light sources, results in larger estimation error, as seen in Figure 6.4. The SHLP estimator outperforms the PLS method, resulting in a more accurate lighting estimation for any number of additional light sources. In fact, the SHLP method estimations are significantly better for lighting complex scenes (RMS error of 0.358 vs. 0.572 for ten additional light sources).



Figure 6.5: SHLPE and PLSE lighting estimation comparison. Left Column: Ground truth scene illuminated by a known spherical harmonics coefficients. Center Column: Scene illuminated by the PLSE method. Scene illuminated by the SHLPE method.

Figure 6.5 shows a visual comparison of both methods. The comparison takes into account the lighting estimations of a scene rendered with known illumination for both techniques. For complex lighting scenarios, the SHLP method estimates more believable lighting settings than the PLS method, note that the SHLP method's estimations resemble the ground truth images while the PLS method produces inaccurate hard shadows and darker ambient light.

The visual perception of the correct lighting setting provided by the SHLP estimator significantly improves the immersion by correctly blending the hands and the virtual environment.

Chapter 7

Environment Light-Probe Estimator

The Environment Light-Probe (ELP) estimator is utilized to recognize the real-world environment lighting, leveraging this lighting information to virtual environments, allowing more convincing lighting composition for XR experiences. The ELP estimator explores spherical harmonics functions to encode the environment lighting into a compact and expressive representation, similar to the SHLP estimator presented in the previous chapter. This strategy allows representing smooth arbitrary area lighting, not limited to a few point light or directional light sources [51].

The ELP estimator model is based on the NAS net convolutional neural network architecture [120] and is capable of predicting the spherical harmonics environment lighting operating over a low-dynamic-range (LDR) image and the user's rotational orientation. The neural network operates over the tree color channels of the LDR image, producing plausible environment lighting arrangements concerning direction, color, and intensities of the light sources.

Real environment panoramas captured in a great variety of places and lighting settings are utilized to produce realistic lighting scenarios. Mixed-reality-views are created by processing the panoramas to produce images similar to those captured by the HMD camera in the runtime of XR applications. The processing of an HDR panorama into mixedreality-views is described with more details in Section 7.1.

The lighting settings in a given panorama can be derived by a projection of spherical harmonics functions [92]. This operation produces a set of SH coefficients. The data-set, which is composed of mixed-reality-view and SH coefficients, is employed for training our lighting estimation neural network, as described in Section 7.1.2. An evaluation o the ELP estimator is performed in Section 7.2 by a set of quantitative and qualitative

experiments. Furthermore, an application that relights XR environments is presented, accompany a comparison of the ELP estimator method and other state-of-the-art approaches for lighting estimation.

7.0.1 Environment Lighting Estimation

The environment lighting is the outcome of a complex combination of physical interactions between light sources and surfaces of objects in the scene. There is a variety of representations that can be used to represent the environment lighting, ranging from simplified models (*e.g.* directional light models) to more complex and physically accurate models (*e.g.* resulting radiance from global illumination algorithms such as radiosity and photon mapping [95]).

In real-time applications, environment lighting is commonly represented by environment maps, which allows image-based lighting (IBL) [14] to be implemented in the rendering process. A single environment map can use high-resolution HDR images, in the order of 8192×4096 pixels. However, a lower sampling rate can produce a good quality environment lighting.

The environment map can be a panorama picture of the real-world environment and is referred to as a light-probe image. The light-probe image is used in this work as a source to produce the mixed-reality-views.

A viable approximation of a light-probe image is the usage of spherical harmonics functions. Any signal can be approximated by projecting it onto spherical harmonics basis functions (Section 2.2). In particular, low-order functions are sufficient to approximate the environment lighting due to the diffuse nature of the signal [92]. The ELP estimator uses this approximation to represent the estimated environment lighting.

The second-order spherical harmonics functions are capable of representing the diffuse environment lighting in a compact format, with only nine coefficients per color channel, and enables rendering techniques such as pre-computed radiance transfer functions [102]. Furthermore, a supposition was made that learning a low-dimensional representation of lighting is a more manageable task than a complex high-dimensional function.

7.0.2 Lighting Estimation CNN Architecture

The architecture comprises a convolution-based feature extractor and three head-networks predicting the RGB lighting coefficients.

The recent advances in deep learning are allowing the development of highly optimized neural networks that are capable of learning complex tasks with a reduced number of parameters. A current trending in deep learning is the use of machine learning methods to automate and design deep neural network architectures. This methodology is called AutoML [38].



Figure 7.1: ELP estimator: architecture overview

The NASNet is a state-of-the-art neural network for image classification and object detection [120], and it is designed by an AutoML process called Neural Architecture Search (NAS). The resulting outcome of the AutoML process is an architecture with state-of-art accuracy and fewer parameters when compared to equivalent neural network
architectures. The ELP estimator uses the NASNet-A [120] as the main convolutional feature extractor. We choose to use the NASNet-A network based on the performance of the state-of-the-art convolutional neural networks available at the time of development of the ELP estimator; the NASNet-A achieved superior performance with a reduced number of parameters when compared with similar neural network models [120].

Since the ELP estimator operates on color images, the model must be able to estimates the lighting contributions for each color channel, this property influences the neural network architecture design. Hence, we model our network as a siamese network [33], where each color channel feeds a NASNet feature extractor. A hypothesis that the feature extractor should work similarly across all the color channels is made. Consequently, a shared weight design like a siamese network could lead to an effective result with a reduced computational cost.

The output of the feature extractor is concatenated with the user's rotational orientation, achieved either from the HMD or from the rotation matrix in the data-set label during training. The concatenated features represent the per-channel latent space features of our mixed-reality-views.

Preliminary empirical observations led to concluding that providing rotational cues to the neural network improves the resulting lighting estimation. The complex interactions between light sources and object surfaces generate intricate global illumination effects. Numerous different lighting settings can produce very similar outcomes, especially when seen through a limited field of view. This ambiguity causes difficulty in the optimization problem of estimating the environment lighting. We suppose that providing a user's rotational orientation can reduce this ambiguity during the training of our neural network.

The ELP method splits the lighting estimation problem into a per color channel scheme, where each one receives separated latent space features accompanied by a prediction head. The prediction heads are connected to their respective latent space features and comprehend a fully connected (FC) network. Those FC networks are identical across color channels, although they do not share weights.

The prediction head is composed of three consecutive FC layers with hyperbolic tangent activation function and dropout regularizer [106]. The first two FC layers have 1024 neurons in the hidden layers, while the last one has 512 neurons. The final layer of each prediction head is an FC layer with nine neurons, accounting for each lighting coefficient.

The ELP estimator models our solution with a regression of the 3×9 spherical har-

monics lighting coefficients. The total loss function of our network is determined by the sum of the loss for each channel, thus:

$$L = L_R + L_G + L_B. ag{7.1}$$

Since we are modeling the problem with second-degree spherical harmonics functions, we can separate them by three bands. Let $k \in R, G, B$ be the index that identifies each lighting channel. The band 0 with coefficient C_k^0 corresponds to the ambient light term, a constant value across all the environment. Band 1 with coefficients C_k^1 correspond to lighting lobes aligned to horizontal, vertical, and depth axis. The terms of band 2, C_k^2 , correspond to the remaining five lighting lobes constituting multiple combinations of directions for each one of the channels. Each coefficient is defined independently for each channel. We attribute a weighted scalar to each band in the loss function. Therefore, the loss function per channel results in:

$$L_{i} = \sum_{k \in \{R,G,B\}} \alpha E(C_{k}^{0}) + \beta E(C_{k}^{1}) + \gamma E(C_{k}^{2}),$$
(7.2)

where E(c) is the mean squared error (MSE).

Figure 7.1 illustrates the architecture of our model, including the per channel separation (a), the siamese feature extractor (b), and the prediction heads (c).

7.1 Learning from HDR panoramas

This section describes the complete pipeline to process the input HDR environment panorama into mixed-reality-views and the corresponding environment lighting. The mixed-reality-view (MRV) is a low-dynamic-range (LDR) color image similar to a photograph taken from a camera located in the HMD, capturing an egocentric view of the user's environment. Spherical harmonics coefficients encode an area light model that represents the environment lighting. Those data are used for training the ELP lighting estimation model.

The goal of this input processing pipeline is to generate sufficient data to model arbitrary environmental lighting settings, capturing the implicit casual relationship between the environment appearance and light sources in the scene. This is accomplished via the usage of real-world HDR environment panoramas present in the Laval indoor HDR database [22]. The input processing pipeline generates data with sufficient diversity



Figure 7.2: Mixed-reality-view processing pipeline

regarding the user's position and orientation, surface and materials properties, and lighting characteristics. Hence, delivering adequate training data for the lighting estimation model.

The usage of HDR panoramas is fundamental to the lighting estimation process. The high-dynamic-range images are used to generate a Spherical Harmonics representation of the environment that captures all the lighting information of the scene. The ultimate goal of our neural network is to learn the HDR SH representation from conventional LDR images.

7.1.1 Laval Indoor HDR Database

For lighting estimation purposes, it is essential to consider a wide range of light sources with distinctive intensities. Low-Dynamic-Range (LDR) images are widespread and highly available, such as photographs captured by consumer cameras or smartphones. However, LDR photographs can not capture in the same picture the brightest spot of a lamp and the dark details of a shadowing area.

The Laval indoor database consists of a large set of high-resolution indoor panoramas, captured in High-Dynamic-Range (HDR). The panoramas capture the entire Field Of View (FOV), with the azimuthal angle of 360 degrees.

There are 2142 panoramas captured at the resolution of 2048×1024 pixels. The scenes have a variety of lighting settings ranging from artificial light sources (ceiling, wall, and table lamps) to natural ones (open window, glass door).

7.1.2 Mixed-reality-views

The required pipeline to achieve the mixed-reality-views from the HDR panoramas aims to mimic the behavior of a camera walking through the panorama environment. To accomplish this, we rotate the panorama horizontally and vertically, simulating the user's head movements in an XR environment. We choose a random vertical ϕ and horizontal θ angle in the range of [-15,15] and [-180, 180] degrees, respectively. We simulate a camera by a perspective projection of the environment panorama. The projection and camera settings were tailored to replicate the camera (HTC Vive integrated camera) employed in our tests.

To approximate the spatial changes in the camera position, we use a warp operation T defined by:

$$T(v,\beta) = \frac{2v_z \sin(\beta) + \sqrt{(-2v_z \sin(\beta))^2 - 4(v_x^2 + v_y^2 + v_z^2) \sin(\beta)^2 - 1}}{2(v_x^2 + v_y^2 + v_z^2)}, \qquad (7.3)$$

where v is the point (v_x, v_y, v_z) , and β is the angle between the camera nadir and the center of projection in the image. This warp operation is based on the warp operation proposed by Gardner et al. [22]. We choose β to correspond to the lowest point in the mixed-reality-view.

A simple gamma-correction process [93] is applied to generate the LDR image. The resulting LDR image is a mixed-reality-view with a rectified crop of the original image and a limited field of view. To obtain the lighting setting of the mixed-reality-view, we project second-order spherical harmonics functions into the rotated panorama. The projection generates nine coefficients, one for each function of the second-order spherical harmonics. Since we intend to estimate the color of the lighting environment, we project a set of nine coefficients for each color channel, resulting in 27 coefficients.

Additionally, we store the rotation matrix that produced the mixed-reality-view, since we consider this additional information useful for the network prediction and is accessible at runtime through the HMD device. The tuple [(mixed-reality-view, rotation matrix), SH coefficients] is the sample of our training data-set. We repeat this operation 64 times for each panorama, totaling 137.088 samples in our data-set. Figure 7.2 illustrate our data-processing pipeline.

7.2 Results and Experiments

In this section, we show the results of our method and discuss the XR applications that are made possible by our lighting estimation method.

7.2.1 Performance

An application was implemented using the Unity Engine for rendering and simulation. The application runs on a desktop computer with an Intel Core i5 CPU clocked at 3.8 GHz and 8 GB of RAM. The model inference was implemented using the TensorFlow library without any runtime specific optimization. The average inference time for a single image on CPU was 523 milliseconds, while on a P100 GPU, we achieved an inference time of 28.3 milliseconds, thus satisfying a real-time constraint, required for an XR application.

For training purposes, an NVIDIA DGX-1 machine was used. The training was executed on $4 \times$ P100 GPUs. We train the lighting estimation model among 9 epochs, which achieved a loss convergence within 4 hours and 27 minutes.



7.2.2 Lighting Estimation

Figure 7.3: Training and validation loss for the Environment Light Probe estimator.

The mixed-reality-view dataset was split into training, validation, and test sets. The splitting ratio utilized was approximately 70% of the data for training (95.936 samples), 25% for validation (34.272 samples), and the remaining 5% for testing (6880 samples). We ensure for our split sets that a mixed-reality-views generated from a specific panorama

appears only in a single set; thus, the split produces disjoint sets. Figure 7.3 shows the training/validation history of the ELP estimator. The hyperparameters used for training was: 64 examples as batch size, 10⁻⁶ learning rate, and the Adam optimizer. The ELP model was trained until convergence of the validation loss achieving an RMS error of 0.0045 on epoch 5.

Table 7.1: ELP estimator: the Root Mean Square Error (RMSE) and R2 score for the lighting estimation in the test dataset. 25%, 50% and 75% correspond to the 1st quartile, median and 3rd quartile.

	RMSE	R2 Score
mean	0.0019	0.9772
std	0.0086	0.0989
25%	0.0002	0.9906
50%	0.0003	0.9970
75%	0.0009	0.9976

Table 7.1 shows the test results of our lighting estimation model. We experimented on images that were never seen by the neural network during the training phase (test set). We use a root mean squared error (RMSE) and R2 score metrics. We obtain the ground-truth SH coefficients from the projection of the HDR panoramas on the SH basis. The RMSE and R2 scores are calculated comparing the ground-truth and the predicted SH coefficients. Figure 7.4 shows examples of the prediction and the ground-truth, represented by Irradiance Environment Maps [92]. The normalized difference highlights the color-wise mean difference in the prediction. Most of the prediction error is related to the small color difference between ground-truth and predicted results.



Figure 7.4: Environment Light Probe estimator prediction results. The input for our model (left) is a limited field-of-view image of the scene. The ground-truth lighting is obtained from an HDR panorama. The prediction image is obtained from the rendered scene using the SH coefficients predicted by the ELP estimator. The difference image is the normalized difference between ground-truth and prediction image.



Figure 7.5: Environment Light Probe estimator comparison with state-of-the-art. The input (left) is a limited field-of-view image of the scene. The ground-truth lighting is obtained from an HDR panorama. The prediction image is obtained from the rendered scene using the SH coefficients predicted by the ELP estimator. Gardner et al. [22] prediction is a coarse environment map of the scene. For comparison, the lighting estimation results of Gardner et al model were we projected on SH basis, and the projected results were used to render an irradiance environment map.

Figure 7.5 compares the ELP model with the state of art methods. The difference image between predicted and ground truth provides a visual cue of the estimation results. The ELP method results visually resemble the ground-truth in lighting color, intensity, and position. While the ELP architecture tries to match the scene color separately, resulting in more vivid colors (high amount of green, red, yellow, and blue colors), Gardner et al. method tend to predict a mean color environment (closest to grey/brown color). Even when predicting considerably different lighting, our model still produces plausible environment lighting.



Figure 7.6: Environment Light Probe estimator on mixed reality scenes. Stock photos with virtual objects lit by our lighting estimation model.

7.2.3 Scenes Relight

A relighting application consists of changing the illumination according to a given lighting setting. Figure 7.6 shows the usage of our lighting estimation method in a relighting context. We estimate the environment lighting in a given scene and relight the virtual objects with the estimated environment lighting.

To demonstrate the generality of our model, we relight stock photos gathered on the internet, introducing 3D objects in the scene (e.g. see Figure 7.6). In this case, we provide an identity matrix as input for the user's orientation. Figure 7.7 shows our method in an outdoor scene. Although only indoor images were used in the CNN training, our network is robust to estimate lighting for indoor and outdoor scenes.



Figure 7.7: Relighting of outdoor scenes. Virtual objects (Horse and lion statue, black crow, and bird) are inserted in an outdoor stock photo. The virtual objects was relighted with our lighting estimation method.

Our method is capable of estimate lighting for indoor and outdoor scenes, However, since the prediction takes account of the whole scene lighting in a single compact representation, localized lights and occlusion in lighting areas poses as a challenging environment for our estimation model, Figure 7.8 shows the undesired result in a relight example. Problems related to the occlusion of the light can be seen in the lion statue on the right of the figure, that is clearly visible in the shadows; while the two lion statues in the back-left portion of the image blend with the environment, the lion in the front-left do not receive

the correct lighting. We plan to investigate this issue in the future by developing a CNN that can learn spatio-temporal changes in the lighting across the scene.



Figure 7.8: Environment Light Probe estimator limitations: localized lights and occlusion in lighting areas can be perceived in mixed reality scenes.

7.3 Discussion

In this chapter of the thesis, a new real-time environment lighting model was developed. The ELP estimator is able to compute plausible environment lighting for XR applications directly from mixed-reality-views with no former constraints. Differently from previous approaches, the method does not rely on any constraints on the scene geometry and nor requires the use of physical probes.

The environment lighting produced is encoded as 3×9 spherical harmonic coefficients (9 for each color channel) predicted by a NAS neural network architecture with a new proposal of latent space, that combines color channel derived features with rotational cues. It was empirically identified that the use of user orientation could effectively enhance lighting prediction by reducing the ambiguity that relates lighting settings to the final

outcome (*e.g.*, different lighting settings can produce the same illumination results).

Our training dataset is defined by a set of LDR mixed-reality-views computed from the Laval indoor panorama dataset. We produce a new dataset from the original one by varying user orientation and camera position with respect to each panorama. While camera rotation was simulated by rotating the panoramas horizontally and vertically, the camera spatial variation was obtained by a warping approach. The final LDR dataset was computed by mapping the HDR images using standard gamma correction.

The experiments have shown that we can produce plausible environment lighting representations without any strong requirements on the inputs. Compared to some stateof-art works, we produced smaller RMSE and R2 values when comparing the predicted environment lighting and the ground-truth. We also show that our method can be easily applied to XR and relighting applications.

The product of the ELP model enables XR applications to change and adapt the environment lighting, allowing realistic lighting and immersive simulations. However, some results regarding scenes with predominantly localized lighting settings can be enhanced. The same can be said about scenes with many occluded objects. This is still a drawback of the ELP method, which can be explained by the fact that we do not rely on any geometrical information about the scene. We intend to pursue this in future works. Moreover, we believe that the impact of the HDR to LDR mapping on the prediction of the lighting representation must also be more deeply investigated in the future.

7.3.1 Limitations and Future Work

This work leverages XR experiences by enabling plausible, consistent, dynamic environment lighting. This is accomplished by a machine learning model that learns the realworld environment lighting through a series of HDR panoramas. Our model takes into account spatial changes of the camera in the environment through a warp operation; however, indoor environments can have highly localized lighting settings that our model can not estimate accurately. Furthermore, since no geometrical information of the scene is known, problems can arise in areas with object occlusion. While our model does not tackle these issues, it can enable future refinements of lighting estimation methods that handle localized lighting and occluded regions.

Chapter 8

Final Remarks

This thesis creates consistent lighting in mixed reality applications through real-world lighting estimation. We developed deep learning estimators to estimate the lighting using three different environment lighting representations. The results show that using a deep learning approach to solve the ill-posed lighting estimation problem is a robust and encouraging approach.

8.1 Mixed Reality Framework

The mixed reality framework presented in this thesis describes a suitable methodology to solve the illumination mismatch problem through the usage of lighting estimation models. The general mixed reality framework describes a methodology to adapt the virtual world environment lighting to a matching real-world environment lighting. The process contains a lighting estimation step that should be executed in real-time during the mixed-reality application runtime. The framework takes as input an image from the real-world scene and is executed continuously while the application is running. Additionally, we also define a specialization of the general mixed reality framework called the hand-based mixed reality framework. This specialization makes use of the human hands light-probe hypothesis that consists of utilizing the human hands as light-probe of the environment lighting. The hand-based mixed reality framework assumes this hypothesis by utilizing the image of the user's hands in the mixed reality environment as the light-probe of the real-world environment lighting.

The methods developed for lighting estimation presented in this thesis shows that it is possible to estimate the environment lighting using a single low-dynamic range image of the real-world environment, making then suitable for the mixed reality framework. We also show that the PLS and the SHLP lighting estimation results contribute to the human hands light-probe hypothesis, conclusively establishing that the human hand contains enough information to make an estimation of the environment lighting. Furthermore, the usage of computer-generated graphics to create a dataset suitable for the training process of the convolutional neural network is examined in the PLS and SHLP methods. The experimental results for both methods show that a convolutional neural network trained on synthetic data is robust enough to generalize the results against real-world data.

8.2 Point-Light Source Estimator

The PLS method estimates the environment lighting of the real-world environment using discrete configurations of lighting settings. The lighting settings in the PLS method consists of point-light sources distributed uniformly on the surface of a sphere around the user's position. The results of the PLS show that this lighting representation is adequate for representing the environment lighting for mixed reality applications. Moreover, the PLS method utilizes the hand-based mixed-reality framework to achieve plausible and consistent environment lighting in mixed-reality applications.

The PLS lighting estimation model is based on a residual convolutional neural network that treats the lighting estimation as a classification task. The trained model achieves a high average classification accuracy (83.15 % for the 25 lighting settings model) while maintaining a reasonable inference speed for mixed reality applications (16 ms). We show through qualitative and quantitative comparations that a level of discretization consisting of 25 lighting settings is sufficient to represent the primary light source of the environment. We also conclude concerning incorrect classifications in the PLS method, that even when the method incorrectly classify an image, the PLS produces visually consistent images by classifying a lighting setting that is in the neighborhood of the ground-truth correct classification. In this case, the resulting mixed-reality scene produced by the PLS method is similar to the ground-truth mixed-reality scene.

8.3 Spherical Harmonics Light-Probe Estimator

The SHLP method estimates the environment lighting of the real-world environment using functions in a spherical harmonics basis. The environment lighting is represented in a light-probe format based on the spherical harmonics lighting. This compact representation allows complex and variable lighting settings in an area-light source model that produces extensive variations in the lighting scenarios regarding the number of light-sources, spatial position, and light intensity. The SHLP utilizes a single set of nine SH coefficients to represent the lighting settings. Despite the fact that this representation does not account for lighting chromaticity, the results of the SHLP shows that this lighting representation is adequate for describing the lighting in complex scenarios where more than one principal light-source is present in the environment. The SHLPE is an evolution of the PLS regarding representativity, allowing more lighting settings to be accurately represented.

The SHLP lighting estimation model is based on a dense convolutional neural network that treats the lighting estimation as a regression task. The trained model produces a low mean error (RMSE of 0.057) associated with the regression of spherical harmonics lighting coefficients. The SHLP method maintains a reasonable inference speed for mixed reality applications (18 ms), enabling real-time lighting estimation. We show for synthetic scenarios through qualitative and quantitative tests that the SHLP lighting estimation method outperforms the PLS method as we increase the number of simultaneous light sources in the scene. The estimation error of the SHLP is relatively lower than the PLS for more than one simultaneous light source, concluding that the SHLP is more suitable for complex lighting scenarios.

8.4 Environment Light-Probe Estimator Method

The ELP method estimates the environment lighting of the real-world environment using the basis of spherical harmonics functions. The environment lighting assumes a lightprobe format based on the spherical harmonics lighting. The ELP utilizes three sets of nine SH coefficients to represent the lighting settings (one set for each color channel). Hence, the method does account for lighting chromaticity. The results of the ELP show that this lighting representation is adequate for representing the lighting in complex real-world scenarios, including light-sources present in indoor and outdoor environments. The ELP introduces a methodology to produce mixed-reality views based on real-world panoramas. The methodology account for projection and spatial variance thought the usage of a spatial warping operation. Furthermore, the mixed-reality view represents the environment lighting with a low-dynamic-range image and a spherical harmonic lighting coefficient extracted from a high-dynamic-range panorama, therefore providing data for the neural network to learn the relation between LDR image and HDR lighting setting.

The ELP estimator shows that a convolutional neural network is capable of learning the environment lighting with minimum information regarding the environment and without assumptions about the environment or prior knowledge regarding the environment geometry or reflectance properties. The ELP lighting estimation model is based on a custom neural network that uses the NAS-A network as a feature extractor. The architecture makes use of channel-wise separation through the usage of a siamese neural network. The ELP treats the lighting estimation as a regression task, resulting in the regression of 27 spherical harmonics lighting coefficients, being nine coefficients for each color channel. The trained model produces a low mean error (RMSE of 0.0019), associated with the regression of the spherical harmonics lighting coefficients. The ELP method maintains a reasonable inference speed for interactive mixed reality applications (28 ms). We show that the ELP is suitable for lighting estimation through real-world scenarios by comparing the results with ground-truth data, obtained from HDRI panoramas. We also conclude that the ELP method is suitable to produce believable mixed reality scenes for relighting applications of real-world footages with the insertion of lighting consistent virtual objects.

8.5 Highlights

The key results and contributions of this thesis are listed below:

- **Deep Learning for lighting estimation** The lighting estimation models developed in this work shows that using deep learning techniques to solve the ill-posed problem of lighting estimation is a viable and robust approach. Moreover, we show that deep learning approaches produce models with low inference time. This characteristic make deep learning models suitable for lighting estimation in real-time applications, such as mixed-reality applications.
- Hand based lighting estimation models In this work, we test the hypothesis of using a human hand as a light-probe of the environment lighting. Two hand based lighting estimation models were developed. The methods provide a consistent environment lighting in egocentric mixed reality applications, effectively solving the illumination mismatch problem.
- **Environment lighting estimation** The environment light-probe estimation model shows that it is possible to estimate the environment lighting without assumptions about the environment or prior knowledge of the scene.

Generalization of synthetic data to real-world We show that it is possible to train a neural network using computer-generated graphics to infer the lighting estimation on real-world data, producing results with low associated error and visually plausible results.

8.6 Future Works

Future works include the focus on investigating novel ways to solve the Spatio-temporal variation in the environment lighting and faster inference methods for mixed-reality applications. The investigations on light-probe sampling techniques and neural network architectures have the potential to capture spatially-varying light-probes in the input image, effectively alleviating the limitations of the developed lighting estimation models.

Manually acquiring light-probe images from the real-world environment is unpractical. The development of an automatic method to acquire such light-probes is an interesting topic of research that could enhance the methods developed in this thesis by providing a richer dataset in the training of the lighting estimation models.

The development of neural network architectures that posses specialized layers capable of learning the spatial variance in a more effective way than current state-of-art networks could lead to a dramatic improvement regarding the spatial localization of the environment lighting. This topic of research is especially crucial for the mixed-reality area in indoor environments, where depth information is usually not available, and numerous light occlusions occur. Furthermore, improvements in the network architecture to provide temporal relations between the data have the potential to increase the accuracy and smoothness of the lighting estimation process for real-time simulations.

References

- AGGARWAL, C. C. Neural Networks and Deep Learning: A Textbook. Springer, aug 2018.
- [2] AITTALA, M. Inverse lighting and photorealistic rendering for augmented reality. The Visual Computer 26 (2010), 669–678.
- [3] BERTSEKAS, D. P.; TSITSIKLIS, J. N. Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3). Athena Scientific, may 1996.
- BLANZ, V.; VETTER, T., ET AL. A morphable model for the synthesis of 3d faces. In Siggraph (1999), vol. 99, pp. 187–194.
- [5] BLINN, J. F. Models of light reflection for computer synthesized pictures. In Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 1977), SIGGRAPH '77, Association for Computing Machinery, p. 192–198.
- [6] BOOM, B. J.; ORTS-ESCOLANO, S.; NING, X. X.; MCDONAGH, S.; SANDILANDS, P.; FISHER, R. B. Interactive light source position estimation for augmented reality with an rgb-d camera. *Computer Animation and Virtual Worlds* (2015).
- [7] BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010. Springer, 2010, pp. 177–186.
- [8] BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In Proceedings of the 27th international conference on machine learning (ICML-10) (2010), pp. 111–118.
- [9] CALIAN, D. A.; LALONDE, J.-F.; GOTARDO, P.; SIMON, T.; MATTHEWS, I.; MITCHELL, K. From faces to outdoor light probes. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 51–61.
- [10] CALIAN, D. A.; MITCHELL, K.; NOWROUZEZAHRAI, D.; KAUTZ, J. The shading probe: Fast appearance acquisition for mobile ar. In SIGGRAPH Asia 2013 Technical Briefs (2013), ACM, p. 20.
- [11] CHEN, G.; DONG, Y.; PEERS, P.; ZHANG, J.; TONG, X. Reflectance scanning: estimating shading frame and brdf with generalized linear light sources. ACM Transactions on Graphics (TOG) 33, 4 (2014), 117.
- [12] CHOE, J.; SHIM, H. Robust approach to inverse lighting using rgb-d images. Information Sciences 438 (2018), 73 – 94.

- [13] CONDE, M. H.; SHAHLAEI, D.; BLANZ, V.; LOFFELD, O. Efficient and robust inverse lighting of a single face image using compressive sensing. In 2015 IEEE International Conference on Computer Vision Workshop (ICCVW) (2015), IEEE, pp. 226–234.
- [14] DEBEVEC, P. Image-based lighting. In ACM SIGGRAPH 2005 Courses (2005), ACM, p. 3.
- [15] DEBEVEC, P. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In ACM SIGGRAPH 2008 Classes (New York, NY, USA, 2008), SIGGRAPH '08, Association for Computing Machinery.
- [16] DEBEVEC, P.; GRAHAM, P.; BUSCH, J.; BOLAS, M. A single-shot light probe. In ACM SIGGRAPH 2012 Talks (2012), ACM, p. 10.
- [17] DESCHAINTRE, V.; AITTALA, M.; DURAND, F.; DRETTAKIS, G.; BOUSSEAU, A. Single-image svbrdf capture with a rendering-aware deep network. ACM Trans. Graph. 37, 4 (July 2018), 128:1–128:15.
- [18] DOMINGOS, P. A few useful things to know about machine learning. Commun. ACM 55, 10 (Oct. 2012), 78–87.
- [19] DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, 61 (2011), 2121–2159.
- [20] EGGER, B.; SCHÖNBORN, S.; SCHNEIDER, A.; KORTYLEWSKI, A.; MOREL-FORSTER, A.; BLUMER, C.; VETTER, T. Occlusion-aware 3d morphable models and an illumination prior for face image analysis. *International Journal of Computer Vision 126* (2018), 1269–1287.
- [21] GAGGIOLI, A. An open research community for studying virtual reality experience. Cyberpsychology, Behavior, and Social Networking 20, 2 (2017), 138–139.
- [22] GARDNER, M.-A.; SUNKAVALLI, K.; YUMER, E.; SHEN, X.; GAMBARETTO, E.; GAGNÉ, C.; LALONDE, J.-F. Learning to predict indoor illumination from a single image. ACM Trans. Graph. 36, 6 (Nov. 2017), 176:1–176:14.
- [23] GARON, M.; LALONDE, J.-F. Deep 6-dof tracking. IEEE transactions on visualization and computer graphics 23, 11 (2017), 2410–2418.
- [24] GARON, M.; SUNKAVALLI, K.; HADAP, S.; CARR, N.; LALONDE, J.-F. Fast spatially-varying indoor lighting estimation. In *Proceedings of the IEEE Conference* on Computer Vision and Pattern Recognition (2019), pp. 6908–6917.
- [25] GHOSH, A.; CHEN, T.; PEERS, P.; WILSON, C. A.; DEBEVEC, P. Estimating specular roughness and anisotropy from second order spherical gradient illumination. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 1161– 1170.

- [26] GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (2011), pp. 315–323.
- [27] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. MIT press, 2016.
- [28] GOODFELLOW, I. J.; WARDE-FARLEY, D.; MIRZA, M.; COURVILLE, A.; BEN-GIO, Y. Maxout networks. In Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28 (2013), pp. III–1319.
- [29] GREEN, R. Spherical harmonic lighting: The gritty details. In Archives of the Game Developers Conference (2003), vol. 56, p. 4.
- [30] GRUBER, L.; LANGLOTZ, T.; SEN, P.; HÖHERER, T.; SCHMALSTIEG, D. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In 2014 IEEE Virtual Reality (VR) (2014), IEEE, pp. 15–20.
- [31] GRUBER, L.; RICHTER-TRUMMER, T.; SCHMALSTIEG, D. Real-time photometric registration from arbitrary geometry. In 2012 IEEE international symposium on mixed and augmented reality (ISMAR) (2012), IEEE, pp. 119–128.
- [32] GUARNERA, D.; GUARNERA, G. C.; GHOSH, A.; DENK, C.; GLENCROSS, M. Brdf representation and acquisition. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 625–650.
- [33] GUO, Q.; FENG, W.; ZHOU, C.; HUANG, R.; WAN, L.; WANG, S. Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1763–1771.
- [34] HABEL, R.; MUSTATA, B.; WIMMER, M. Efficient spherical harmonics lighting with the preetham skylight model. In *Eurographics (Short Papers)* (2008), pp. 119– 122.
- [35] HADAMARD, J. Lectures on Cauchy's problem in linear partial differential equations. Courier Corporation, 2003.
- [36] HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE* international conference on computer vision (2015), pp. 1026–1034.
- [37] HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (2016), pp. 770–778.
- [38] HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. arXiv preprint arXiv:1908.00709 (2019).
- [39] HE, Z.; SUI, X.; JIN, G.; CAO, L. Progress in virtual reality and augmented reality based on holographic display. *Applied optics* 58, 5 (2019), A74–A81.
- [40] HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on 14*, 8 (2012).

- [41] HOLD-GEOFFROY, Y.; ATHAWALE, A.; LALONDE, J.-F. Deep sky modeling for single image outdoor lighting estimation. In *The IEEE Conference on Computer* Vision and Pattern Recognition (CVPR) (June 2019).
- [42] HOLD-GEOFFROY, Y.; SUNKAVALLI, K.; HADAP, S.; GAMBARETTO, E.; LALONDE, J.-F. Deep outdoor illumination estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (jul 2017), vol. 1, IEEE, p. 6.
- [43] HOSEK, L.; WILKIE, A. An analytic model for full spectral sky-dome radiance. ACM Transactions on Graphics (TOG) 31, 4 (2012), 95.
- [44] HOWELL, K. B. Principles of Fourier analysis. CRC Press, 2016.
- [45] HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; WEINBERGER, K. Q. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (2017), pp. 4700–4708.
- [46] JARRETT, K.; KAVUKCUOGLU, K.; RANZATO, M.; LECUN, Y. What is the best multi-stage architecture for object recognition? In 2009 IEEE 12th International Conference on Computer Vision (Sep. 2009), pp. 2146–2153.
- [47] JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014).
- [48] JIDDI, S.; ROBERT, P.; MARCHAND, E. Reflectance and illumination estimation for realistic augmentations of real scenes. In *IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'16 (poster session)* (2016).
- [49] JIMENEZ, J.; GUTIERREZ, D. Screen-space subsurface scattering. GPU Pro: Advanced Rendering Techniques (2010), 335–351.
- [50] KÁN, P.; KAUFMANN, H. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (2013), IEEE, pp. 133–141.
- [51] KAUTZ, J.; SNYDER, J.; SLOAN, P.-P. J. Fast arbitrary brdf shading for lowfrequency lighting using spherical harmonics. *Rendering Techniques 2*, 291-296 (2002), 1.
- [52] KIM, K.; GU, J.; TYREE, S.; MOLCHANOV, P.; NIESSNER, M.; KAUTZ, J. A lightweight approach for on-the-fly reflectance estimation. In *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2017).
- [53] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2015).
- [54] KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (Nov 2007), pp. 225–234.
- [55] KNECHT, M.; TRAXLER, C.; MATTAUSCH, O.; WIMMER, M. Reciprocal shading for mixed reality. *Computers & Graphics 36*, 7 (2012), 846–856.

- [56] KOLKUR, S.; KALBANDE, D.; SHIMPI, P.; BAPAT, C.; JATAKIA, J. Human skin detection using rgb, hsv and ycbcr color models. arXiv preprint arXiv:1708.02694 (2017).
- [57] KOTZIAS, D.; DENIL, M.; DE FREITAS, N.; SMYTH, P. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2015), KDD '15, Association for Computing Machinery, p. 597–606.
- [58] KURT, M.; EDWARDS, D. A survey of brdf models for computer graphics. ACM SIGGRAPH Computer Graphics 43, 2 (2009), 1–7.
- [59] LALONDE, J.; EFROS, A. A.; NARASIMHAN, S. G. Estimating natural illumination from a single outdoor image. In 2009 IEEE 12th International Conference on Computer Vision (Sep. 2009), pp. 183–190.
- [60] LALONDE, J.-F.; EFROS, A. A.; NARASIMHAN, S. G. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision* (2011).
- [61] LALONDE, J.-F.; EFROS, A. A.; NARASIMHAN, S. G. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision 98*, 2 (2012), 123–145.
- [62] LALONDE, J.-F.; NARASIMHAN, S. G.; EFROS, A. A. What do the sun and the sky tell us about the camera? *International Journal of Computer Vision 88*, 1 (2010), 24–51.
- [63] LE CHÉNÉCHAL, M.; CHATEL-GOLDMAN, J. Htc vive pro time performance benchmark for scientific research. In *ICAT-EGVE* (2018).
- [64] LECUN, Y.; BENGIO, Y., ET AL. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks 3361, 10 (1995), 1995.
- [65] LECUN, Y.; JACKEL, L. D.; BOSER, B.; DENKER, J. S.; GRAF, H. P.; GUYON, I.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine* 27, 11 (1989), 41–46.
- [66] LEGENDRE, C.; MA, W.-C.; FYFFE, G.; FLYNN, J.; CHARBONNEL, L.; BUSCH, J.; DEBEVEC, P. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *The IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (June 2019).
- [67] LETTRY, L.; VANHOEY, K.; VAN GOOL, L. Darn: a deep adversarial residual network for intrinsic image decomposition. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) (2018), IEEE, pp. 1359–1367.
- [68] LI, Z.; SNAVELY, N. Learning intrinsic image decomposition from watching the world. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018), pp. 9039–9048.

- [69] LI, Z.; SNAVELY, N. Learning intrinsic image decomposition from watching the world. In *The IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (June 2018).
- [70] LI, Z.; SUNKAVALLI, K.; CHANDRAKER, M. Materials for masses: Svbrdf acquisition with a single mobile phone image. In *Proceedings of the European Conference* on Computer Vision (ECCV) (2018), pp. 72–87.
- [71] LOMBARDI, S.; NISHINO, K. Reflectance and illumination recovery in the wild. *IEEE transactions on pattern analysis and machine intelligence 38*, 1 (2015), 129–141.
- [72] LOMBARDI, S.; NISHINO, K. Radiometric scene decomposition: Scene reflectance, illumination, and geometry from rgb-d images. In 2016 Fourth International Conference on 3D Vision (3DV) (Oct 2016), pp. 305–313.
- [73] MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (2013), vol. 30, p. 3.
- [74] MAIER, R.; KIM, K.; CREMERS, D.; KAUTZ, J.; NIESSNER, M. Intrinsic3d: High-quality 3d reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *The IEEE International Conference on Computer* Vision (ICCV) (Oct 2017).
- [75] MANDL, D.; YI, K. M.; MOHR, P.; ROTH, P. M.; FUA, P.; LEPETIT, V.; SCHMALSTIEG, D.; KALKOFEN, D. Learning lightprobes for mixed reality illumination. In 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (Oct 2017), pp. 82–89.
- [76] MARQUES, B. A. D.; CLUA, E. W. G.; VASCONCELOS, C. N. Deep spherical harmonics light probe estimator for mixed reality games. *Computers & Graphics* 76 (2018), 96–106.
- [77] MARQUES, B. A. D.; DRUMOND, R. R.; VASCONCELOS, C. N.; CLUA, E. Deep light source estimation for mixed reality. In Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP, (2018), INSTICC, SciTePress, pp. 303–311.
- [78] MARQUES, R.; BOUVILLE, C.; RIBARDIÈRE, M.; SANTOS, L. P.; BOUATOUCH, K. Spherical fibonacci point sets for illumination integrals. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 134–143.
- [79] MARSCHNER, S. R.; GREENBERG, D. P. Inverse rendering for computer graphics. Citeseer, 1998.
- [80] MEILLAND, M.; BARAT, C.; COMPORT, A. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (2013), IEEE, pp. 143–152.
- [81] MEKA, A.; MAXIMOV, M.; ZOLLHÖFER, M.; CHATTERJEE, A.; SEIDEL, H.-P.; RICHARDT, C.; THEOBALT, C. Lime: Live intrinsic material estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018).

- [82] MILGRAM, P.; TAKEMURA, H.; UTSUMI, A.; KISHINO, F. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies* (1995), vol. 2351, International Society for Optics and Photonics, pp. 282–293.
- [83] MITCHELL, T. M. Machine Learning. McGraw-Hill Education, mar 1997.
- [84] NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (2010), pp. 807–814.
- [85] NAM, H.; HAN, B. Learning multi-domain convolutional neural networks for visual tracking. In Proceedings of the IEEE conference on computer vision and pattern recognition (2016), pp. 4293–4302.
- [86] NISHINO, K.; NAYAR, S. K. Eyes for relighting. ACM Transactions on Graphics (TOG) 23, 3 (2004), 704–711.
- [87] PEREZ, R.; SEALS, R.; MICHALSKY, J. All-weather model for sky luminance distribution-preliminary configuration and validation. *Solar energy* 50, 3 (1993), 235–245.
- [88] PESSOA, S. A.; MOURA, G. D. S.; LIMA, J. P. S. D. M.; TEICHRIEB, V.; KELNER, J. Rpr-sors: Real-time photorealistic rendering of synthetic objects into real scenes. *Computers & Graphics 36*, 2 (2012), 50–69.
- [89] PHONG, B. T. Illumination for computer generated pictures. Commun. ACM 18, 6 (June 1975), 311–317.
- [90] PLAYCANVAS. Image-based-lighting. https://developer.playcanvas.com/ en/user-manual/graphics/physical-rendering/image-based-lighting, 2020. [online, accessed 23-March-2020].
- [91] PORTER, T.; DUFF, T. Compositing digital images. In Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 1984), SIGGRAPH '84, Association for Computing Machinery, p. 253–259.
- [92] RAMAMOORTHI, R.; HANRAHAN, P. An efficient representation for irradiance environment maps. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (2001), ACM, pp. 497–500.
- [93] REINHARD, E.; WARD, G.; PATTANAIK, S. N.; DEBEVEC, P. E.; HEIDRICH, W. High Dynamic Range Imaging - Acquisition, Display, and Image-Based Lighting (2. ed.). Academic Press, 2010.
- [94] RICHTER-TRUMMER, T.; KALKOFEN, D.; PARK, J.; SCHMALSTIEG, D. Instant mixed reality lighting from casual scanning. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (Sept 2016), pp. 27–36.
- [95] RITSCHEL, T.; DACHSBACHER, C.; GROSCH, T.; KAUTZ, J. The state of the art in interactive global illumination. *Computer Graphics Forum 31*, 1 (2012), 160–188.

- [96] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature 323*, 6088 (1986), 533–536.
- [97] RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal* of Computer Vision (IJCV) 115, 3 (2015), 211–252.
- [98] SHAHLAEI, D.; BLANZ, V. Realistic inverse lighting from a single 2d image of a face, taken under unknown and complex lighting. In 2015 11th IEEE international conference and workshops on automatic face and gesture recognition (FG) (2015), vol. 1, IEEE, pp. 1–8.
- [99] SHAHLAEI, D.; PIOTRASCHKE, M.; BLANZ, V. Lighting design for portraits with a virtual light stage. In 2016 IEEE International Conference on Image Processing (ICIP) (2016), IEEE, pp. 1579–1583.
- [100] SHI, S.; HSU, C.-H.; NAHRSTEDT, K.; CAMPBELL, R. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proceedings of the 19th ACM international conference on Multimedia* (2011), ACM, pp. 103–112.
- [101] SLATER, M.; KHANNA, P.; MORTENSEN, J.; YU, I. Visual realism enhances realistic response in an immersive virtual environment. *IEEE Computer Graphics* and Applications 29, 3 (May 2009), 76–84.
- [102] SLOAN, P.-P.; KAUTZ, J.; SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans. Graph. 21, 3 (July 2002), 527–536.
- [103] SMISEK, J.; JANCOSEK, M.; PAJDLA, T. 3d with kinect. In Consumer Depth Cameras for Computer Vision. Springer London, 2013, pp. 3–25.
- [104] SONG, S.; FUNKHOUSER, T. Neural illumination: Lighting prediction for indoor environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019), pp. 6918–6926.
- [105] SPEICHER, M.; HALL, B. D.; NEBELING, M. What is mixed reality? In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (New York, NY, USA, 2019), CHI '19, Association for Computing Machinery.
- [106] SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDI-NOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
- [107] SUN, T.; BARRON, J. T.; TSAI, Y.-T.; XU, Z.; YU, X.; FYFFE, G.; RHEMANN, C.; BUSCH, J.; DEBEVEC, P.; RAMAMOORTHI, R. Single image portrait relighting. ACM Transactions on Graphics (TOG) 38 (2019), 79.
- [108] SUTTON, R. S.; BARTO, A. G. Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series). A Bradford Book, nov 2018.

- [109] TSUMURA, N.; DANG, M. N.; MAKINO, T.; MIYAKE, Y. Estimating the directions to light sources using images of eye for reconstructing 3d human face. In *Color and Imaging Conference* (2003), vol. 2003, Society for Imaging Science and Technology, pp. 77–81.
- [110] WALTON, D. R.; STEED, A. Dynamic hdr environment capture for mixed reality. In Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology (2018), ACM, p. 18.
- [111] WANG, H.; LIN, S.; YE, X.; GU, W. Separating corneal reflections for illumination estimation. *Neurocomputing* 71, 10-12 (2008), 1788–1797.
- [112] WEBER, H.; PRÉVOST, D.; LALONDE, J.-F. Learning to estimate indoor lighting from 3d objects. In 2018 International Conference on 3D Vision (3DV) (2018), IEEE, pp. 199–207.
- [113] WEBER, H. J.; ARFKEN, G. B. Essential Mathematical Methods for Physicists, ISE. Elsevier, 2003.
- [114] WHELAN, T.; SALAS-MORENO, R. F.; GLOCKER, B.; DAVISON, A. J.; LEUTENEGGER, S. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research* 35, 14 (2016), 1697–1716.
- [115] ZARIT, B. D.; SUPER, B. J.; QUEK, F. K. Comparison of five color models in skin pixel classification. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on* (1999), IEEE, pp. 58–63.
- [116] ZHANG, E.; COHEN, M. F.; CURLESS, B. Emptying, refurnishing, and relighting indoor spaces. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016) 35, 6 (2016).
- [117] ZHANG, J.; SUNKAVALLI, K.; HOLD-GEOFFROY, Y.; HADAP, S.; EISENMAN, J.; LALONDE, J.-F. All-weather deep outdoor lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).
- [118] ZHOU, Y.-T.; CHELLAPPA, R. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks* (1988), vol. 1998, pp. 71–78.
- [119] ZOLLHÖFER, M.; THIES, J.; GARRIDO, P.; BRADLEY, D.; BEELER, T.; PÉREZ, P.; STAMMINGER, M.; NIESSNER, M.; THEOBALT, C. State of the art on monocular 3d face reconstruction, tracking, and applications. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 523–550.
- [120] ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8697–8710.