Light Source Estimation via Intrinsic Decomposition for Novel View Synthesis

Master Thesis

Author(s):
Tetruashvili, David Simon

Publication date:
2023

Permanent link:
https://doi.org/10.3929/ethz-b-000645817

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Light Source Estimation via Intrinsic Decomposition for Novel View Synthesis

Master's Thesis

David Simon Tetruashvili
Department of Computer Science

Advisors: Prof. Dr. Martin Ralf Oswald
Dr. Partha Das
Erik Sandström

Supervisor: Prof. Dr. Luc van Gool
Computer Vision Laboratory
Department of Information Technology and Electrical Engineering

29th November 2023

Abstract

In this thesis, we propose a method for light source estimation via intrinsic decomposition within a Neural Radiance Field-like setting. We begin by exploring a learned method to recover light sources from intrinsic components - reflectance, shading, and surface normals. As demonstrated, the learned mapping is unable to properly learn the relationship between the different intrinsic components, and instead overfits some components while ignoring the rest. However, inference of local SH lighting, while somewhat spatially discontinuous, shows promise for future extensions. For these reasons, we later pivot our approach to directly optimize global spherical harmonics lighting, in which the in-scene radiance is represented by coefficients of a single global basis function. With this optimization-based approach, we successfully reconstruct a coarse estimate of the lighting of our scenes. To validate and support our method, we introduce a novel synthetic dataset, which consists of intrinsic components of several scenes. For each scene in our dataset, we provide geometry, shading, reflectance, and fully diffuse and rendered passes. We hope that extensions of our work could serve as modules of future NeRF models, enabling the joint optimization and refinement of lighting, geometry, and radiance.

Acknowledgements

I would like to thank my supervisors Martin Oswald and Partha Das for their excellent support, despite it taking place remotely over video-conference. I thank Erik Sandström for putting the considerable effort he put in to facilitating my remote supervision and diving deep into the intricacies of this work alongside myself. Additional thanks to Mike Roberts for providing his data wrangling code, which saved time during exploration.

I would also extend my thanks to the proofreaders of this manuscript: Julian Koller, Jordan B. Seligmann, Jackson Stanhope, and Martina Roncoroni. Additional thanks, again, to Jordan B. Seligmann for the pointing out of many crucial bugs in the knick of time. To Jackson Stanhope for placing myself on track for this project, his expertise in computer vision and graphics, and his apt literature suggestions. And to Julian Koller for his near-constant words of encouragement and support. Finally, I would like to thank Ma'ayan Ben Simon for keeping me sane all this time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Light Source Estimation is the task of recovering a description of the lighting parameters of a scene from images. This work explores the viability of the inclusion of light source estimation into a Neural Radiance Field representation of a scene.

A Neural Radiance Field (NeRF) [20] implicitly represents a scene via a neural network. When presented with a set of posed images (views), NeRF approximates the plenoptic function, induced by a scene via a simple multi-layer perceptron (MLP). The plenoptic function describes the flow of light through space in all directions [3]. Since this approximation generalizes over scene-space, NeRFs excel at the task of novel view synthesis – generating views not seen in the input. The MLP maps 5D spatial and directional coordinates to a volume density and view-dependent emitted radiance at that spatial location.[20] The volume density field corresponds to the scene's geometry.

Intrinsic Image Decomposition (IID) is the task of disentangling reflectance and shading from an image. [1] Reflectance is independent of the lighting of the scene and corresponds to the colour of the objects in it. Shading is a function of the geometry and lighting conditions of the scene. Recent works by Ye et al.[36] as well as Choi, Kim and Kim[37] apply IID within a NeRF setting. This grants us access to novel view synthesis of disentangled reflectance and shading.

Since NeRFs learn the geometry of the scene, knowing an estimate of the intrinsic components gives us a path to concurrently estimate the lighting conditions as well, giving us the potential of a lighting condition-aware NeRF system in future work.

This work is of interest to us because most NeRF representations attempt to optimize entangled signal of radiance. A decomposed representation theoretically has a lower degree of freedom compared to standard NeRFs. A lower DOF parameterization — one which seeks lighting, reflectance, and geometry separately — is typically easier to optimize, which can lead to better results. Knowledge of the lighting conditions of a neural radiance field would enable downstream tasks and improvements. For one, a joint optimization of lighting, reflectance, and shading (therefore radiance as well) within a NeRF would allow on-the-fly refinements of the field's geometry, which is currently subject to noise. Additionally, a decomposed representation would also enable scene editing and relighting.

1.2 Focus of this Work

The main goal of this thesis is to propose a method of light source estimation within a pipeline based on intrinsic components which systems like [36, 37] can provide.

To this end, we construct the Intrinsic Dataset, an intrinsically decomposed dataset. We take Intrinsic Dataset as a proxy output of an intrinsic NeRF model. Additionally, we create tooling which can be used to extend the dataset with a new scene with a small amount of tweaking.

With the samples from the Intrinsic Dataset, we explore whether we can directly learn a mapping from intrinsic components to lighting representations. We first attempt to predict the dominant incident light direction from the object surface in single-light-source scenes. We expand this model to predict an omni-directional local spherical harmonics representation of light, capable of representing multiple lights.

We adapt these to formulate a direct gradient-based optimization to find global lighting conditions which best explain the scene. We implement this optimization with the intention that an extension of it will become an additional module of a future NeRF system for light estimation.

Please note that we do not anchor our light source estimation to the scene-space, and neither do we tackle modelling non-Lambertian effects in the scene. Both of these aspects would be natural extensions to our method and are relegated to future work.

In summary, our contributions are as follows:

- The Intrinsic Dataset which holds intrinsically decomposed views of single-object scenes and tooling its expansion.
- Exploration of neural mappings from intrinsic components to lights representations.
- Direct optimization of global lighting via intrinsic decomposition.

1.3 Thesis Organization

In Chapter 2 we present the theoretical background as well as relevant extensions of four related tasks of this work – Neural Radiance Fields, Light Source Estimation, Intrinsic Image Decomposition, and Spherical Harmonics lighting. Chapter 3 motivates and presents a technical overview of the Intrinsic Dataset – an intrinsically decomposed dataset we have created to act as an in-place proxy of systems like [36, 37]. We give a description of our methods in Chapter 4, where we first describe our proof-of-concept methods and then give a description of our direct optimization of SH lighting on the Intrinsic Dataset. Chapter 5 presents and gives an analysis of the hyperparameter sweeps we have carried out for the proof-of-concept methods, and presents the experiments on the components of the direct optimization. Finally, Chapter 6 presents the results of our experiments and ablation studies on direct optimization. We show renderings of the test splits of the Intrinsic Dataset using found lighting. Finally, in Chapter 7 we give a conclusion and provide an outlook on extensions and future work.

Chapter 2

Related Work

Since the popularization of Neural Radiance Fields [20] as a neural scene representation framework, many works sought to extend the technique. The breadth of these works ranges from fidelity enhancements [25, 27, 41], to scene editing capabilities such as relighting [22, 39]. More recently, NeRFs gained extensions such that they could be applied to *inverse rendering* – obtaining scene descriptions from rendered images [35, 29] and *intrinsic image decomposition* – splitting the image into intrinsic layers; obtaining an image representation which disentangles reflectance, shading, and geometry from the render [36, 37]. These developments open opportunities for further extensions, such as harnessing the aforementioned intrinsic decomposition methods in solving *light source estimation* tasks within this paradigm.

Neural Radiance Fields.

Neural Radiance Fields are implicit scene representations which use simple neural networks – multi-layer perceptrons (MLPs) – as universal approximators to learn the scene's plenoptic function [20]. The plenoptic function is a 5D function which describes the amount of light which flows through every spatial point in every direction in the scene. Therefore, a NeRF stores a representation of the density of light and its radiance within the weights of its core network, taking a collection of posed images of the scene as input.

Here, the end goal is to produce a pixel colour which would be seen from a given spatial location in some viewing direction. To do so, [20] propose training a network to map continuous 5D (spatial and directional) coordinates to radiance values as well as the occupied volume density. Integrating these along a ray gives a predicted pixel colour for that ray. In this way, NeRFs treat each spatial point as a light source with an intensity corresponding to the density at that point. NeRFs use the *Photometric loss*, the mean squared error between the predicted colour and the corresponding colour from a posed image, to train the network.

Since a NeRFs approximates the plenoptic function from a sample of known posed images, it gains the advantage that it can preform novel-view synthesis. This task attempts to reconstruct/render the scene from some view not yet seen or given, which we can do by querying the NeRF from that view.

Since the primary tool driving NeRFs are simple MLPs, the framework lends itself well to extensions. [27, 25] extend internal sampling methods to increase graphical fidelity. [41] attempts to quell 'floater' artefacts emerging from ambiguities within the reconstruction. [28, 28, 30] remove the need for lengthy training by replacing back-propagation with direct optimization of a space-filling field data structure such as a voxel grid.

On top of works extending NeRF itself, more recent works delve into solving related tasks using a NeRF-based core, like *intrinsic image decomposition* and *light source estimation*. We discuss further methods in the following sections for each of these.

Light Source Estimation.

The task of light source estimation seeks to retrieve the lighting conditions of a scene given image(s) or view(s) of it. The task is strongly coupled with the choice of representation of light. For example, assuming an infinitely distant light source present in all directions requires only to approximate its intensity [33, 40], while a local spatial representation of finite-extent point or area sources calls for spatial clustering [11]. The task is also heavily affected by the assumptions on light transport, since some lighting condition effects which may be present in a view can explicitly break these. [23]

Light source estimation has been tackled within multiple paradigms. However, we will focus heavily on leveraging neural representations. [7, 32, 16, 15, 24] take the neural approach and provide lighting estimates by feeding single (depth and colour) images through their networks. Each of these methods carries its assumption on the representation of light – [15] in particular, assumes the number of lights present as a hyperparameter. [33] uses differentiable rendering to create synthetic OLAT (One-light-at-a-time) samples of existing images to establish a prior for novel lighting conditions, and jointly optimizes for the scene's materials and lighting environment map. [35], like [33], uses joint optimization of materials, geometry, and lights, and applies multi-view photometric stereo to a NeRF model, and therefore requires multiple OLAT samples per posed view as input.

Learning an implicit mapping between images and lighting conditions is far from the only approach. By assuming that specularities on objects in the scene can be accurately modelled by mirror-like materials, we can cast bundles of rays into the scene from these in a supposed direction of a light. [11] bounces ray cones from specularities brighter than a set threshold within a SLAM process, and clusters a space-filling voxel grid by the number of intersections with these rays. Those voxels which count many intersections are then good candidates for light source locations. [32] uses mirror-like specularities to reduce ambiguity during the optimization, something that is avoided in other neural methods by the diversity and quantity of input views through photometric consistency.

In our work, we explore learning a neural representation of light in section 4.2, and we optimize for a lighting representation in section 4.3.

Intrinsic Image Decomposition.

Intrinsic Image Decomposition is a computer vision task which aims to retrieve two intrinsic layers of an image, namely the reflectance and shading.[26] According to [31, 2] we can express image formation under the Lambertian assumption as

$$\mathbf{I} = m(\mathbf{n}, \mathbf{l}) \int_{\omega} \rho(\lambda)\, e(\lambda)\, f(\lambda)\, \mathrm{d}\lambda \tag{2.1}$$

where $\mathbf{I}$ is the image and λ is the incoming light wavelength from the visible spectrum ω. m is a function of scene geometry and lighting; $\mathbf{n}$ denotes the surface normal and $\mathbf{l}$ the direction of incoming light. e describes the power distribution of the light source, and f indicates the sensitivity of the camera over the visible spectrum. ρ is the reflectance in the scene, and is related to the albedo or colour of the objects within it.

When we work with pixel images, we can consider each pixel to be a quantity already integrated through eq. (2.1) and simplify the image formation equation to

$$\mathbf{I} = \mathbf{R} \odot \mathbf{S} = \mathbf{R} \odot \Psi(\mathbf{N}, \mathfrak{L}) \tag{2.2}$$

where $\odot$ is the element-wise product, $\mathbf{R}$ is the reflectance image (the colour of the object) and $\mathbf{S}$ is the shading image. $\mathbf{S}$ is analogous to the m term in eq. (2.1), and therefore, stems from the geometry $\mathbf{N}$ and lighting $\mathfrak{L}$ of the scene, and thus can be decomposed into them. We can use a rendering function Ψ to do the opposite and compute shading from geometry and lighting.

In the simplest case, when we know a singular direction of a light $\mathbf{l}$ and the surface normal vector of a point $\mathbf{n}$, the shading function boils down to the dot product between the two bounded at zero:

$$\Psi(\mathbf{n}, \mathbf{l}) = \max\{\mathbf{n} \cdot \mathbf{l},\, 0\}. \tag{2.3}$$

From eq. (2.2) we can also say that shading acts as a scaling term on reflectance, but emerges from how much of that reflectance we can see due to the interplay of geometry and light.

While geometry can still be described in the 2D image domain (a normal vector at each point), lighting usually cannot since it is tied to the 3D space the scene occupies. Therefore, recovering it is relegated to inverse rendering and light source estimation tasks.

[22] focuses on disentangling lighting from geometry for the purposes of relighting, however, relies on that the initial conditions are known. [29] further disentangles scene into visibility, material properties, and reflectance, and does not rely on known initial lighting. [36] constructs losses and constraint penalties on shading, albedo components, deriving these as outputs of a spatial MLP, applying intrinsic decomposition in a NeRF setting. Since the decomposition is not done on a per-image basis, the paradigm is more granular than classical IID, since we can decompose individual pixels or bundles of pixels. Most recently, [37] (published shortly before the completion of this thesis) foregoes employing costly Monte-Carlo integration methods to obtain a reconstruction loss and instead relies on pre-filtered radiance layers to reduce reconstruction to a network query.

Neural representation decomposition methods provide us with access to tackle the light source estimation task within an end-to-end neural field setting. We use this development as inspiration for our work, assuming the existence of a neural representation capable of accurately decomposing a scene into reflectance, shading, and geometry. We use the Intrinsic Dataset described in chapter 3 as a proxy for such a system.

Spherical Harmonics Shading and Lighting.

Shading is an arbitrary piecewise continuous function $f_s : \mathbf{N} \times \mathfrak{L} \mapsto [0, 1]$ without cast-shadows and is governed by the orientation of the surface to the direction of incoming light. Defining it over the surface of a (unit) sphere is convenient since the domain acts as the space of normals $\mathbf{N}$ – that is, orientations – on which we can impose lighting $\mathfrak{L}$ to get f_s. A set of *Spherical Harmonics* (SH) Y_l^m forms a complete orthonormal set of functions on the surface of the unit sphere — a linear combination of these approximates f_s with arbitrary precision given a high enough order l. [28, 18, 33] use SH to represent low-frequency lighting, and hence evaluate shading. Following their example, in chapter 4 we do the same.

As is, a given $Y_l^m : S^2 \mapsto \mathbb{C}$ is a complex polynomial function, however, for the purposes of expressing purely real functions like shading, we may define a real spherical harmonics basis. We express it depending on the sign of the degree m of each individual harmonic and combine the harmonic's real and imaginary parts:

$${}_{\Re}Y_l^m = \begin{cases} \sqrt{2}\,\Im\left[Y_l^{|m|}\right] & \text{if } m < 0 \\ Y_l^0 & \text{if } m = 0 \\ \sqrt{2}\,\Re\left[Y_l^m\right] & \text{if } m > 0 \end{cases} \tag{2.4}$$

Since $l \geq 0$ and $-l \leq m \leq l$, we have that a set of harmonics of order l (that is, set of all harmonics up to and including those of order l) has $(l+1)^2$ components. Consequently, this means that to define some function on S^2 via SH of order l, we must supply a set of real-valued coefficients $\mathbf{L} \in \mathbb{R}^{(l+1)^2}$. Approximating $\mathfrak{L}$ and in turn f_s boils down to finding an appropriate $\mathbf{L}$.

Given such a set $\mathbf{L}$ and a region shaded by the function it represents, we may evaluate the resultant shading by supplying the normal of the surface $\mathbf{n} = (n_x, n_y, n_z)$ to a rendering function [4]:

$$\Psi_{\text{SH}}(\mathbf{n}, \mathbf{L}) = \sum_{l'=0}^{l} \sum_{m'=-l'}^{l'} {}_{\Re}Y_{l'}^{m'}(\mathbf{n})\mathbf{L}_{l'm'} \tag{2.5}$$

where $\mathbf{L}_{l'm'}$ is the coefficient of the corresponding harmonic. For convenience, we often write $\mathbf{L}_i$ for $0 \leq i \leq (l+1)^2$ to denote a corresponding coefficient (assuming an order on the set of harmonics).

Since lower-order harmonics only have the capacity to represent low-frequency smooth lighting, we can assign $\mathbf{L}^{(j)}$ per subdivision of space to represent local lighting. A fine-grained and sparse voxel-based approach is shown in [28] whereas [12] opt to fully subdivides space, while [18] operates on images and outputs per-pixel local SH lighting. This gives rise to spatially varying spherical harmonics (SVSH) which overcomes the shortcomings of assuming a single global lower-frequency light. Spherical harmonics of higher orders exhibit so-called ringing artefacts [13], so the literature consensus is to represent signals such as lighting or shading by second-order SH.

Since physically plausible lighting is always a strictly non-negative function, so is f_s. Some authors offer extensions to find such $\mathbf{L}$ which ensures $\Psi(\mathbf{n}, \mathbf{L}) \geq 0 \, \forall \, \mathbf{n} \in S^2$. [18] proposes a direct non-negativity constraint as an additional optimization loss term, while [23] formulates a measurement matrix M such that $\Psi(\cdot, M\mathbf{L}) \geq 0$ which represents the same constraint. Another approach would be to introduce a normalization transformation to map the $\mathbf{L}$-resultant function to $[0, 1]$. These additions are explored in section 4.3.2.

Chapter 3

Datasets

As part of this work, we propose a small object-based NeRF-style dataset which contains ground truth shading, reflectance, and Lambertian-only passes, in addition to lighting meta-data. In section 3.1 we present our rationale for the creation of the Intrinsic Dataset and define the specification of the desired features and properties. We give a thorough overview of the contents of the dataset created by our generation tooling in section 3.2, while in section 3.3 we detail the implementation of these tools as well as their technical requirements. Section 3.4 describes an expansion on the Intrinsic Dataset which enables on-the-fly generation of point-lit one-light-at-a-time (or OLAT) image/pixel samples from already existing samples. Finally, section 3.5 gives a brief discussion of the limitations of our generation tooling.

3.1 Motivation

While there already exist datasets designed for use in intrinsic decomposition tasks, these are not necessarily what a typical NeRF-like dataset would be. NeRF datasets typically contain many overlapping and posed views of all surfaces in the scene [38], for example, taken from frames of a video recording. We require a dataset which contains such views, and moreover, we need the intrinsic decompositions of each view. Furthermore, to facilitate light source estimation, we would like the dataset to contain meta-data describing the lighting of each scene. We would also benefit if we have the ability to modify aspects of the scenes, like the lighting, to extend the dataset.

For example, HyperSim [21] is a popular dataset which includes photorealistic and intrinsically decomposed scenes with numerous lights per scene. However, it does not have a description of these lights. Neither does it allow for easy editing of the scene nor its lighting, since the raw 3D scene representations are not provided. [21] also requires additional pre-processing to be done to its raw image data before use such as tonemapping, and while it does contain many posed views per scene, these are not guaranteed to be continuous or comprehensively show all surface points of the scene. The combination of these factors would conceivably complicate the development of our method.

To have a NeRF-style dataset with the additional features we want, we have created a simple object-based and intrinsically decomposed dataset based on [20] which we will refer to as the Intrinsic Dataset from now on. In this chapter, we will further motivate, present, and overview of its components, and describe its creation. The combination of the dataset and its tooling serve as the main data resource throughout the development of this work.

To carry out development of our method in incremental steps, we opt to create tools for intrinsically decomposing existing Blender scene representations. With this approach, we gain the advantage of expanding the dataset by applying the generation tools to new scenes when needed.

Figure 3.1: Overview of the Intrinsic Dataset showing several samples from the `test` split of the Chair scene. From top to bottom: Full colour render, Diffuse render conforming to the Lambertian assumption, Albedo, Shading (excluding shadows and self-shadowing), Camera-space normal, and Depth map passes. Samples which focus on auxiliary objects, not shown for clarity.

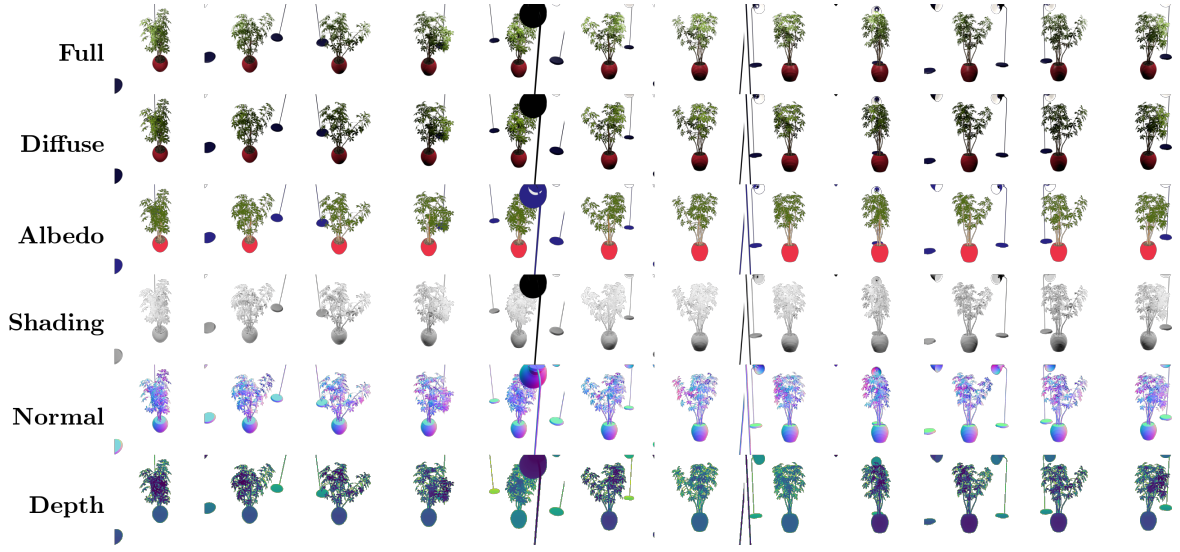

Figure 3.2: Overview of the Intrinsic Dataset showing several samples from the `test` split of the Ficus scene. The layout of this figure is identical to fig. 3.1.

Note on Terminology: For the purposes of describing the dataset, we refer to the set of all image passes given a set of camera parameters as a 'view' in the Intrinsic Dataset and any single image is referred to as either a 'pass' or simply 'image' associated with that view. Later on, when discussing the generation of OLAT samples, and especially, when discussing the dataset in the context of using it for neural model training, the term 'sample' or 'pixel-sample' is used to refer to a set of all passes of a single pixel in the dataset for any given image-sample.

We find it is important to first lay out the requirements of the Intrinsic Dataset and give their general justifications. Since the Intrinsic Dataset should be useful for developing methods for light source estimation via intrinsic decomposition, the dataset should include intrinsically decomposed attributes of contained images. For development and evaluation purposes, it is desirable for the dataset to contain versions of each view lit individually by each of the present lights in the scene, so called one-light-at-a-time or OLAT samples/images. As explained above, we would also like to have additional variations of contained samples, for example, versions of the images lacking specular reflection or non-diffuse effects overall. This enables us to fix that assumption in place and evaluate the method both when it holds and when it does not. Furthermore, the ability to apply the point-light assumption to these OLAT samples further reduces the complexity of the sample, allowing for incremental increases in sample complexity that the method could be evaluated on.

To enable the use of the dataset as a mock-up of a NeRF-like novel view synthesis model's output, it must also contain enough varied samples to train a such a model. Basing the estimate from datasets used to train various flavours of NeRF or Plenoxels [28] we would need the Intrinsic Dataset to contain about 100 samples per scene. We also desire that these views sufficiently overlap, and collectively show all relevant surfaces and objects in the scene. In a similar vein, for the purposes of applying machine learning methods to this dataset, we would want to generate training, testing, and validation sample subsets for each scene.

Moreover, we would require meta-data associated with each sample in the set. In particular, the poses and calibrations of the cameras used to generate the samples, as well as meta-data on the ground-truth lighting. These two pieces of meta-data along with the depth buffer would enable reconstruction of the scene in space. In the end, a desired dataset should have the following for each scene:

1. A comprehensive number of views $\mathbf{I}_i$ given by camera poses $\mathbf{C}_i$ collectively containing all objects in the scene; each surface point should be represented in at least several dataset pixel-samples.

2. For each view $\mathbf{I}_i$ present in the dataset:

    - The associated intrinsic decomposition, the components of which are:
        - The reflectance maps $\mathbf{R}_i$ carrying the chromatic content of the material in the overall view.
        - The shading $\mathbf{S}_i$ map, which entangles the light information with the geometry of the scene.
    - A diffuse version of the non-decomposed view $\mathbf{I}_i^{(\text{diff})}$.
    - A representation of non-scale-invariant associated geometry in the form of a normalized depth buffer $\mathbf{D}_i$ and the surface normals $\mathbf{N}_i$.
    - The camera parameters $\mathbf{C}_i$.

3. The description of the lighting, namely of each light $\mathfrak{L}_j$.

3.2 Intrinsic Dataset Overview

The Intrinsic Dataset is based on the NeRF Object Dataset [20] as this dataset is widely used and fits our desired attributes of relative scene simplicity while still capturing enough geometric complexity to serve as

a proxy for more complex scenes. The general workflow for the creation of the Intrinsic Dataset is to amend existing scenes from [20], intrinsically decompose them, and derive additional meta-data from the scene representation.

In the development of this work, two of the scenes from [20] were selected to be intrinsically decomposed and included into the Intrinsic Dataset. However, as we will present in section 3.3 the dataset can be easily expanded by generating/rendering samples from 3D scene representations, so long as the scene conforms to a few requirements. The scenes chosen for this work were the 'chair' and 'ficus' scenes, since these do not predominantly display non-Lambertian effects and have significantly varied geometry.

In total, for each included scene, the Intrinsic Dataset contains 100 randomly selected views with the camera tracking towards the origin in the training and validation splits, and 200 views sampled sequentially along a spiral trajectory in the testing split (according to code provided in [20]). In addition to these, for any added auxiliary objects in a scene, we sample 20 views with each of these off-centre objects as the subject. We do not include these views in the testing splits. Overall, the Intrinsic Dataset contains 480 samples for the Chair scene, 520 samples for the Ficus scene.

We number each view $\mathbf{I}_i$ and each pass is named with a corresponding suffix. We compose each view of the following passes (see a selection of samples shown in fig. 3.1 and fig. 3.2):

- Combined (also called 'full') render pass $\mathbf{I}_i^{(\text{comb})}$ which shows any non-Lambertian effects present in the scene.
- Diffuse render pass $\mathbf{I}_i^{(\text{diff})}$ which shows a simplified version of the combined pass excluding non-Lambertian effects while keeping the two images as similar as possible.
- Reflectance pass $\mathbf{R}_i$ which shows the underlying diffuse colour of each material in the scene.
- Shading render pass $\mathbf{S}_i$ which shows the shading due to geometry in the scene. This pass excludes any shadows or self-shadowing.
- Camera-space normal render pass $\mathbf{N}_i$ which uses a common approach to render surface normals in camera-space as pixels.
- Depth buffer pass $\mathbf{D}_i$ which stores the depth values along the positive z-axis (in the direction of the camera) normalized to a defined range (between 0 and 8 units by default).

All passes are saved as RGBA images, where the alpha matt determines pixel-occupancy by an object. The normal passes are stored as PNG-16 images, while the rest are PNG-8. Compression settings were set to a minimum or turned off. For each of the splits in each scene, we record the camera parameters $\mathbf{C}_i$ in a JSON file. Since the camera used has a square frame, we only record the horizontal field of view angle (`camera_angle_x`) from which we later can derive focal length and construct the intrinsics matrix K. The extrinsics of the camera are captured per view via its camera-to-world (`c2w`) transform matrix and stored in the `frames` list along with its respective view number.

For each scene, we record the properties of the light present in a separate JSON file. For each light, we record its type (either point light or area light), its world transform matrix, colour (in RGB), and brightness via its energy. When collecting meta-data about area lights, we additionally store the light's extent in the x and y directions.

Finally, we also store the random number generator seed that was used to generate the data, so that the camera placements, for example, could be reproduced (see section 3.3.3).

```
chair_intrinsic
├── train
│   ├── r_xxx.png
│   ├── r_xxx_albedo.png
│   ├── r_xxx_depth.png
│   ├── r_xxx_diffuse.png
│   ├── r_xxx_normal.png
│   ├── r_xxx_shading.png
│   ├── ...
│   └── seed.json
├── test
│   └── ...
├── val
│   └── ...
├── lights_chair_intrinsic.json
├── transforms_train.json
├── transforms_test.json
└── transforms_val.json
```

Figure 3.3: Example of the file structure of the Intrinsic Dataset.

3.3 Generating the Intrinsic Dataset

To create the Intrinsic Dataset we adopt the existing meta-data formats and relative file structure from [20] to be able to reuse existing dataset parsers (such as those available in [38]). We use the source Blender scene files as a basis for our render passes. We make a few alterations to the scene files before generating decomposed samples for [20] which are detailed below. The samples themselves are generated using an extension of the camera-posing logic used to create [20]. The extended script is responsible for four tasks: a) Determining render settings such as resolution, b) Establishing the compositing node tree for all render passes (see section 3.3.2), c) Sampling the camera extrinsics per view, and d) Collecting the meta-data for each view. Each pass is then rendered in turn. An additional script collects the meta-data related to scene lighting.

3.3.1 Alterations to NeRF Object Dataset Scene Representations

Simplifying Materials and Textures

Some materials in the original [20] use blended BSDFs which deviate too much from their diffuse approximation (e.g., velvet). We first manually change the materials. We do this in a way that preserves as much of the original appearance, but in doing so, gain the ability to toggle non-Lambertian effects via Blender's render layers feature. This usually entails adding a diffuse blend layer to non-diffuse materials, which only becomes visible once the non-diffuse effects are toggled off. Other small deviations from [20] include the change of some materials from having near-black reflectance to a brighter value to avoid instabilities during gradient-based optimization. For example, the mostly black specular plant pot in the Ficus scene was changed to a bright red colour to be able to differentiate the shading on it more clearly.

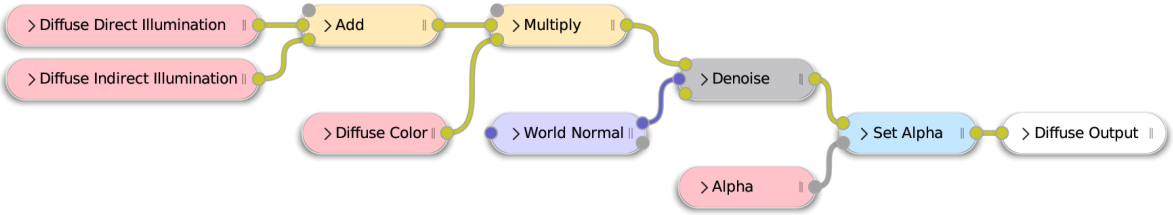

Figure 3.4: Compositing of the Diffuse pass $\mathbf{I}_i^{(\text{diff})}$.

Insertion of Local Lighting

In addition to the main object in each scene (the chair and the ficus respectively), additional objects were added to represent in-scene local lighting. Two or three large overhead lamp object meshes [10] were inserted into the scene, and an area light consistent with their placement and size was added for each lamp mesh. These were added as visible and interpretable references of the lighting conditions of each scene. The number of lights was chosen to be 2 for the chair scene and 3 for the ficus scene, while we set their locations arbitrarily within a short distance of the main object/origin. The orientations were hand-picked to point in the general direction of the origin and provide varied lighting across the object. For each light, we record the position and orientation in space via its transform matrix, its brightness, colour, and type (area light or point light). For each scene, we record a meta-data file containing this information for each light present, which we store in the main directory of each scene (see fig. 3.3).

3.3.2 Compositing Render Passes

Each pass is rendered without a background and only using the locally inserted lighting without a use of the environment emitter included in [20]. Default de-noising settings are kept, with each render taking 256 samples-per-pixel unless stated otherwise. We selectively composite the various Render Layers which Blender automatically produces during the overall render process to generate the necessary render passes. Note that passes which carry physical properties, specifically the normal and depth passes, are generated without colour management post-processing (with the view transform set to 'Raw').

Diffuse and Reflectance Passes

The diffuse and albedo passes are created by leveraging the `DiffCol`, `DiffDir`, and `DiffInd` Render Layers, which store the material colour, direct illumination, and indirect illumination for each pixel in the image and compositing them together as shown in Figure 3.4. While the diffuse pass uses the rendering eq. (3.1) to combine all three render layers, the reflectance pass is only a composite of the `DiffCol` with the alpha matte.

Shading Pass

Similar to the diffuse and albedo passes, the shading pass is rendered by compositing the `DiffDir` Render Layer with a greyscale version of the `DiffInd` Render Layer. However, this pass is rendered separately after all lights present are toggled to not cast shadows. That way we get a clean shading pass uninterrupted by self-shadowing or occlusions.

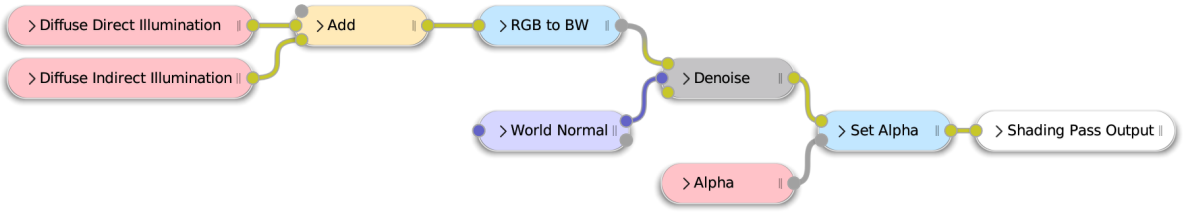

Figure 3.5: Compositing of the Shading pass $\mathbf{S}_i$ (with lighting set to not cast shadows).

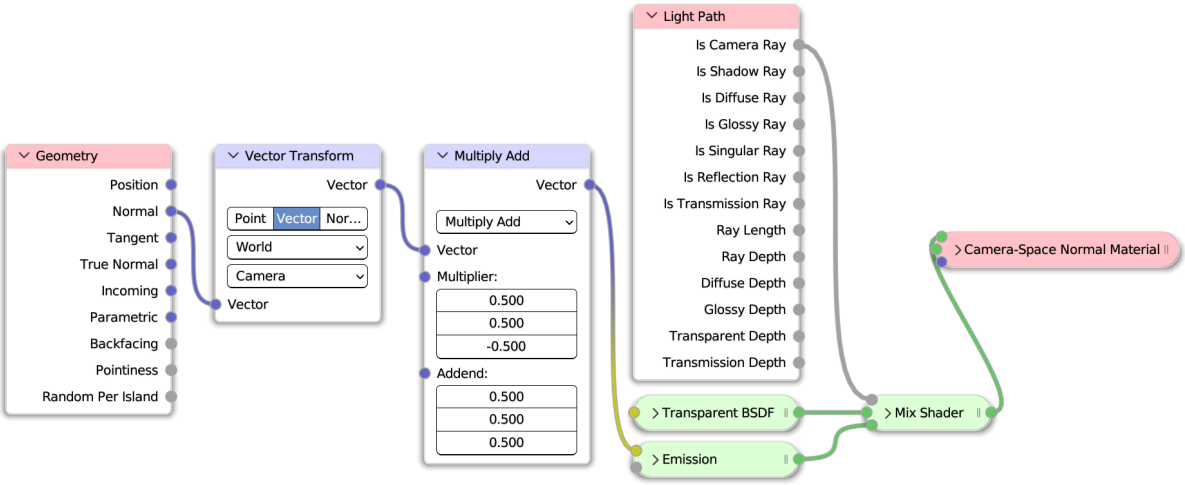

Figure 3.6: Shader Node tree used to create a camera-space normal surface material.

Normal Pass

While Blender does provide a material which visualizes camera-space normals via the *Matcap* material list, it is considered to be inaccurate as it uses a bitmap lookup texture instead of computing the normals from the geometry of the mesh [19]. For this reason, we implement a 'camera-normal material' (shown in Figure 3.6) based on [19] which computes the pixel colour from the world-space normals taken from the `Normal` Render Layer. These are then normalized to prevent numerical inaccuracies and transformed to camera-space. The vector coordinate values are then shifted and scaled to be within the RGB colour range. Note that the mapping assumes negative values for the z-components of the normals, since the camera-space z-axis is taken to be positive in the direction of the camera.

In order for the 'camera-space material' to not be affected by any lighting present in the scene, we treat any primary ray intersecting scene geometry as carrying an emission of the computed colour. Any other light path is set to a transparent material. In this way, only primary rays carry the encoded colour to the output pass and all other light paths are ignored.

Finally, we apply this material non-destructively as a global override to all objects present in the scene, such that the normal passes $\mathbf{N}_i$ could be rendered.

Depth Pass

Note on RGB to Depth conversion: Although the depth passes are generated and stored in the same way as [20], we amend them slightly in software later in the pipeline. The depth passes in [20] assume that brighter

pixel values correspond to pixels closer to the camera, and vice versa. Later on in our pipeline we use the convention that brighter pixel values correspond to higher depth values, i.e., those further away from the camera, and vice versa. We do this convention conversion as this it is assumed by the various dependencies further on in the pipeline.

The depth values themselves are rescaled to 8-bit unsigned integers from the range between 0m and 8m by default, since this range captures the included scenes well. This, by extension, means that to extract the original depth values from the Intrinsic Dataset we also assume this range was used to create the depth passes and use these range limits for rescaling back to concrete depth values.

Combined Pass

After the alterations to the scene representation, the combined pass is rendered using all Render Layers and default render settings – it is Blender's default composited render output.

3.3.3 Compiling Meta-data and Sampling Views

The views are sampled in two modes: either randomly on the upper hemisphere (for the training and validation splits) or sequentially on a spiral (for the testing split). In both cases, the camera points towards either the origin, or to a given location of an auxiliary object (such as a light) and lies a given distance from said object set in a way to capture a significant portion of it.

For splits which require randomness in the sequence of sampled views, we set a seed for the pseudo-random number generator. This is done to maintain consistency between those sets of passes that cannot be generated simultaneously (e.g., colour renders and shading passes for which we disable shadowing) since the scene representations differ. This seed is written to file and stored in the corresponding split's directory. Since we sample passes for auxiliary objects separately, we store this seed separately. Thereafter, if a seed file is present, we read it for subsequent runs of the generation tool.

3.4 Deriving OLAT Samples

Sometimes it is desired to have an image or pixel sample which is lit by a single light source in the scene, like for example in the context of material acquisition or photogrammetry. Though, it might not always be desirable to store these on disk. In this section, we will describe the process of generating OLAT samples via the existing passes which already exist in the Intrinsic Dataset.

Our goal is to create image and pixel samples which satisfy the Lambertian assumption while being lit by single lights already present in the scene. Additionally, we compute these samples as though the lights are point lights, since this is a simple setting which we can use as a proving ground for further methods. A pass which is lit by a light l in this way, we will call $\mathbf{I}_{i,l}^{(\text{OLAT})}$.

To this end, we solve

$$\mathbf{I}_{i,l}^{(\text{OLAT})} = \mathbf{R}_i \odot \mathbf{S}_{i,l}^{(\text{OLAT})} = \mathbf{R}_i \odot (\mathbf{N}_i \odot \Omega_l)\,. \tag{3.1}$$

$\mathbf{S}_{i,l}^{(\text{OLAT})}$ is the shading obtained from the object being lit by the point-light l and Ω_l is the set of light vectors induced by l pointing towards l from the surface points of the object. Note that we cannot use $\mathbf{S}_i$ here, since that pass is not derived from a single point-light, in fact, in our case $\mathbf{S}_i$ contains entangled shading from all area lights in the scene.

We can compute $\mathbf{S}_{i,l}^{(\text{OLAT})}$ per-pixel by computing the corresponding light vectors Ω_l. First, we compute the set of visible surface points $\mathbf{P}_i$ in this view in $\mathbb{R}^3$. To find $\mathbf{P}_i$ we project the depth pass $\mathbf{D}_i$ via the camera intrinsics matrix K and pose it via the camera transform matrix $\mathbf{C}_i$. Finally, we compute vectors between

$\mathbf{P}_i$ and position of the chosen light l which we then normalize. Carrying out this process for each view i and each light position l in the scene gives the full set of singular OLAT samples.

3.5 Limitations of the Intrinsic Dataset

Though we tried to make the generation and expansion of the Intrinsic Dataset as streamlined as possible, there are nonetheless quirks in the generation process. For one, manual editing of the scene it required. In particular, to avoid rendering void pixels in the diffuse pass, we must ensure that all surfaces have at least one purely diffuse material attached. Since we rely on in-scene modelled lighting, we must also manually add these. Another issue in the generation process requires several render jobs on slightly different scene representations. For example, the camera-space normal pass must be rendered separately from all others since we must apply the camera-space normal material globally to the scene, while the shading pass requires us to set all lighting to not cast shadows. While it is technically possible to set this changes programmatically within our generation script, we found this to be unreliable and to cause properties of one pass bleeding over to others, when ruining an entire generation batch. Therefore, the process was left as is.

Since we rely on the depth pass to compute OLAT samples, it is desirable to store depth values in a higher bit-depth format to avoid rescaling and quantization errors, as currently these passes are stored in the PNG-8 image format.

Chapter 4

Methodology

In this chapter, we give a detailed overview of the methods which we have developed as part of this work. We initially present preliminary, proof-of-concept methods. However, the limitations that those methods posed made it necessary to shift our focus to the optimization method discussed in detail in the (chapter 6). We first give a description of how we pre-process the Intrinsic Dataset in section 4.1. Next, we elaborate on the methods used for predicting a dominant light direction with LightMLP (section 4.2.1) and granular SH lighting with LightSH (section 4.2.2). Finally, we present a direct optimization of global SH lighting of the scene in section 4.3.

4.1 Data Preprocessing

It behoves us to describe the common data preprocessing we do as a preliminary step in the methods presented further on in this chapter. The Intrinsic Dataset consists of 800×800 8/16-bit PNG images with transparency. To be consistent throughout our methods, we first transform the images into the canonical range between [0,1] for colour data, and [-1,1] for vector data and store these as 32-bit floating-point values. The depth pass is not transformed. Since the Intrinsic Dataset does not model a background for the included scenes, to avoid training on invalid data samples, we always extract only the foreground or 'occupied' pixels in each image via its depth pass – a pixel is considered occupied if there exists a finite depth value associated with it. Unless stated otherwise, we downscale the images to a size of 400×400. We do so using nearest-neighbour interpolation such that we do not retain interpolation artefacts in non-colour data passes which more advanced interpolation techniques would introduce. Additionally, for directly optimizing global SH coefficients (see section 4.3) we randomly choose half of the available samples to decrease epoch size due to the much smaller dimensionality of the parameter-space. In our method pipeline figures (fig. 4.1 and fig. 4.3) these operations are shown in orange.

4.2 Predicting Light Directions via MLP

Representing lights can be challenging to do in practice. Common representations are specialized for some lighting conditions and cannot generalize to others. A collection of narrow locally placed spotlights cannot be modelled via a coarse environment map, while ambient effects such as outdoor lighting are difficult to capture with a collection of local lights. For example, [23, 23] choose to represent lighting via an environment map, but therefore they lack the ability to effectively model a localized light source. On the other hand, [11] clusters light rays on a voxel grid to pinpoint local point-like light sources, and therefore works well finding desk lamps, but perhaps not a window illuminating a room.

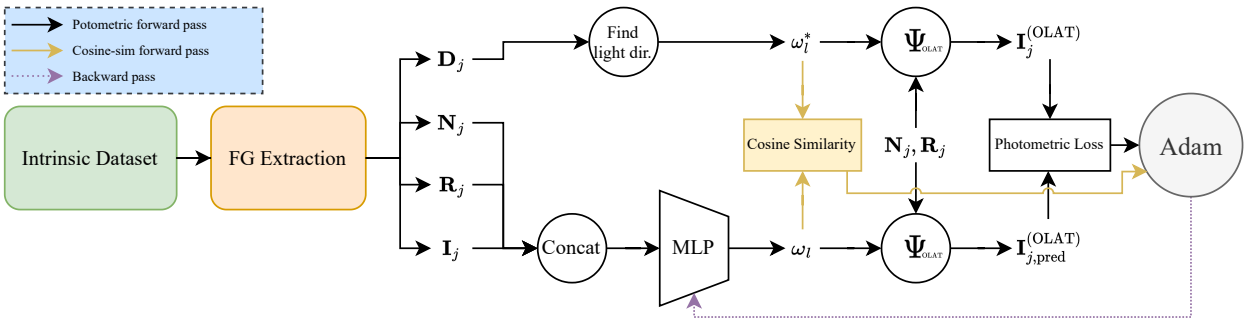

Figure 4.1: Pipeline diagram for the LightMLP which predicts single light direction vectors ω_l after being fed features from Intrinsic Dataset. Concatenated normal, reflectance, and image vectors are inputs. In its directly supervised mode, LightMLP derives its loss via the cosine similarity to a computed ground truth light direction ω_l^* (shown in yellow). A semi-supervised mode of LightMLP renders GT OLAT and predicted OLAT pixel colours and computes the photometric loss between them. Note that the unitary constraint (eq. (4.1)) is not shown for conciseness.

Our initial idea for solving light source estimation within a NeRF-like setting was to construct a neural representation of lighting, leveraging a field formulation similar to NeRF capable of light-representation across scene-space. Our aspiration was to later represent local lights via a spatial MLP. This way we could leverage the potential of end-to-end training in a NeRF-like system and not be limited by a choice of a concrete lighting representation such as an environment map or a voxel grid. To this end, we developed proof-of-concept non-spatial models we dub LightMLP and LightSH for conciseness. LightMLP served as a method to predict a dominant light direction from a surface point, while LightSH predicts per-pixel SH coefficients $\mathbf{L}$ as to be able to capture more complex lighting.

4.2.1 LightMLP

Early stages of this work were concerned with evaluating the effectiveness of a simple neural representation of light. Building from the ground up, LightMLP was intended to show if we can effectively predict the incoming light direction vectors given inputs from Intrinsic Dataset and without providing spatial information. Figure 4.1 shows the pipeline of this network. After loading the Intrinsic Dataset and performing the preprocessing from section 4.1, we concatenate the normal, reflectance, and image inputs $\langle \mathbf{N}, \mathbf{R}, \mathbf{I} \rangle_j$ for an occupied pixel j to feed to an MLP which gives us a predicted light direction vector ω_l in world-space. We project the depth pass $\mathbf{D}_j$ into the scene via the camera pose to find the position of the surface point $\mathbf{x}$ captured by the pixel. From this, and an annotated light position $\mathbf{l}$ in the scene, we find the ground truth light direction $\omega_l^* = \frac{\mathbf{l}-\mathbf{x}}{\|\mathbf{l}-\mathbf{x}\|_2}$ also in world-space. From there we may directly compare the two to derive a loss.

For direct comparison, we use cosine similarity, as we desire that the directions align as much as possible (shown in yellow arrows in fig. 4.1). Alternatively, we can take the two light direction vectors, along with the normal and reflectance at j and use eq. (2.3) to render OLAT pixel colours from both. We then apply the photometric loss to these (shown in black arrows in fig. 4.1). To ensure that we predict valid direction vectors, we use an unitarity constraint, which penalizes the difference of the Euclidean norm of ω_l from one. We add this to our loss function with a strength multiplier λ_u:

$$\mathcal{L}_u = \lambda_u (\|\omega_l\|_2 - 1)^2 \tag{4.1}$$

During training, we compute multiple OLAT versions of each image of the scene – one for each $\mathbf{l}$ annotation – effectively increasing the number of samples by the number of lights in the scene. That way,

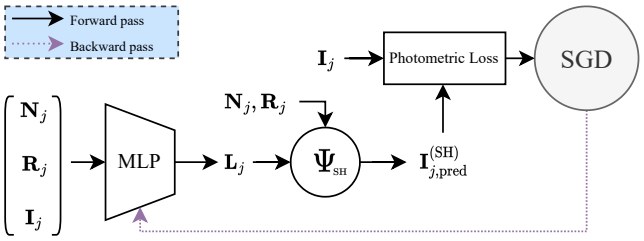

Figure 4.2: Pipeline diagram for LightSH, a variation LightMLP which predicts per-pixel SH coefficients over the image. Concatenated normal, reflectance, and shading vectors are fed into an MLP which predicts second-order SH coefficients, and uses photometric loss between SH-shaded pixel colour and ground truth.

we hope to expand the sample space for the model and avoid overfitting to a signal from a single input feature instead of the interplay between $\langle \mathbf{N}, \mathbf{R}, \mathbf{I} \rangle_j$ and ω_l^*.

4.2.2 LightSH

One big limitation of LightMLP is that it strictly assumes a singular and point-like light in the scene to which all ω_l point towards, something which seldom is the case. It is then also unclear what LightMLP would predict in an unsupervised setting with multiple lights in the scene; although point-lights are known to exhibit additive behaviour, predicting some single average light direction in the case of two point-lights would not solve light source estimation. Later, in chapter 6 we will see that even in single light source scenes, this representation degenerates due to overfitting.

To address this, we extend LightMLP to instead predict second-order spherical harmonics coefficients $\mathbf{L}$ per network query. Not only does this remove the problem with predicting discrete ω_l, we also gain the ability to train LightSH via self-supervision, since ground truth light positions $\mathbf{l}$ are no longer necessary.

Figure 4.2 shows the differences of LightSH w.r.t. LightMLP. For some input pixel j we input the MLP with the features from Intrinsic Dataset and map them to a 9-dimensional output $\mathbf{L}_j \in \mathbb{R}^9$. We render the predicted pixel colour by eq. (2.5) and pass it onto photometric loss.

Similar to LightMLP, the granularity of the prediction space of LightSH allows it to capture non-smooth shading changes in the image. We hope that the weights of the MLP would retain scene lighting information, as not to make the prediction of $\mathbf{L}$ too granular and disconnected on a query to query basis. Although LightSH is more expressive than its predecessor, we still lack an integration method which infers overall lighting conditions of the scene from the individual per-pixel $\mathbf{L}$. In fact, this would be difficult to do without first anchoring the predictions to the scene space.

Although we note that LightSH outperforms LightMLP in reconstructing images of the scene, since it lacks constraints and priors on the output shading and is not given spatial information to, for example, establish shading-smoothness, it still is prone to predicting shading discontinuities along otherwise smooth surfaces. Something that we further discuss in chapter 6.

4.2.3 Implementation Details of LightMLP and LightSH

Both LightMLP and LightSH are implemented using the PyTorch [17] autograd framework. The core of each approach is an MLP with ReLU non-linearities. Each MLP takes in the input features without any additional preprocessing, that is, LightMLP has 12 input nodes while LightSH has 9. Each hidden layer's

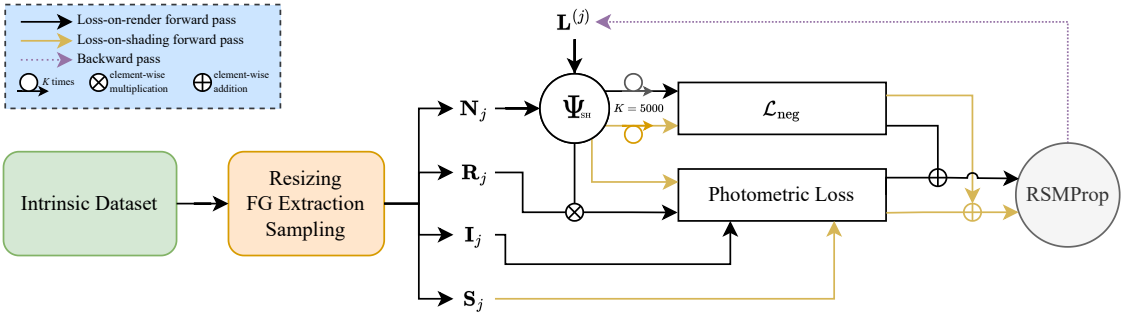

Figure 4.3: Pipeline diagram for directly optimizing for global SH lighting. We compute a predicted pixel colour using the SH coefficients $\mathbf{L}^{(j)}$ and the normal and reflectance inputs via eq. (2.2). $\mathbf{L}^{(j)}$ is penalized for negativity through $K = 5000$ uniformly sampled evaluations of $f_s^{\mathbf{L}}$ over S^2 by the non-negativity constraint $\mathcal{L}_{\text{neg}}$ (eq. (4.5)). We apply the photometric loss between the output of eq. (2.2) and the GT pixel colour $\mathbf{I}_j$, and update $\mathbf{L}^{(j)} \leftarrow \mathbf{L}^{(j+1)}$ with RMSProp. We experiment with a direct loss-on-shading between GT shading and the output of eq. (2.5) (shown in yellow).

size is a constant power of two except the final hidden layer, which is half the size of the previous. We train all three models for 4 epochs of non-downscaled 800×800 images without any further subsampling (i.e., the full dataset).

LightMLP has two modes, a directly supervised mode on the GT light direction vectors ω_l^* and a semi-supervised mode via the photometric loss on OLAT renders of Intrinsic Dataset (induced by ω_l^*). Supervised-LightMLP has 3 hidden layers of sizes [256, 256, 128]. It uses the cosine similarity loss and the unitarity constraint with $\lambda_u = 0.66$. In the semi-supervised mode, LightMLP has 5 layers of size 128 and one of size 64. We still apply the unitarity constraint on the light direction output with $\lambda_u = 0.9$.

Both models were trained using Adam with a learning rate of 4×10^{-2} in the case of supervised-LightMLP and 2×10^{-2} in the semi-supervised mode using batches of size 104 and 32 samples respectively. We apply a 0.2 dropout rate after each layer.

LightSH has 3 hidden layers of sizes [256, 256, 128]. It was trained using SGD with a momentum of 0.9 and a learning rate of 2×10^{-2} using a mini-batch size of 40 samples. We apply a 0.25 dropout rate after every layer.

In chapter 5, we present and discuss the parameter sweeps we executed to find the concrete model parameters for both LightMLP and LightSH.

4.3 Direct Optimization of SH Lighting

Reflecting on the shortcomings of the neural representation in section 4.2.1, we seek to replace the mapping from intrinsic properties to a lighting representation with directly optimizing the representation based on the properties. As presented in chapter 2, SH-based lighting is inherently a parametric representation and, together with the photometric loss, lends itself well to be formulated as an optimization problem. We construct a gradient-based soft-constrained optimization of spherical harmonics lighting. This gives us a global and environmental light which can coarsely capture the scene's lighting through reconstruction. We include a non-negativity constraint from [18, 23] to this basic model to ensure physically plausible lighting. The overview of this pipeline is given in fig. 4.3.

4.3.1 Optimization Setting

We seek to minimize the photometric loss between our SH-rendered pixel colours and members of the Intrinsic Dataset. Taking a stream of samples $\langle \mathbf{N}, \mathbf{R}, \mathbf{I} \rangle$ from the Intrinsic Dataset as input, we want to minimize eq. (4.2) which our lighting representation gives rise to:

$$\mathcal{L}_{\text{PM}}(\mathbf{L} \,|\, \mathbf{R}, \mathbf{N}, \mathbf{I}) = (\mathbf{R} \odot \Psi_{\text{SH}}(\mathbf{N}, \mathbf{L}) - \mathbf{I})^2 \tag{4.2}$$

where Ψ_{SH} is the rendering eq. (2.5). To put another way, the continuous shading function $f_s^{(\mathbf{L})} : S^2 \mapsto \mathbb{R}$ induced by our choice of $\mathbf{L}$ should fit the discrete shading samples $\mathbf{S}$.

The full statement of the optimization problem then becomes

$$\begin{aligned} \hat{\mathbf{L}} = \arg\min_{\mathbf{L}} \quad & \frac{1}{J} \sum_j \mathcal{L}_{\text{PM}}(\mathbf{L} \,|\, \mathbf{R}_j, \mathbf{N}_j, \mathbf{I}_j) \\ \text{s.t.} \quad & \Psi(\mathbf{N}, \mathbf{L}) \geq 0 \quad \forall \mathbf{N} \in S^2 \end{aligned} \tag{4.3}$$

where j is an occupied pixel and J is the total number of these.

We desire that the optimized lighting $\mathbf{L}$ is physically plausible, that is non-negative, so we include the hard constraint on $\Psi_{\text{SH}}(\cdot, \mathbf{L})$ which we formulate as a soft constraint in the next section. In chapter 5, we demonstrate that iterative optimization alone does not suffice in satisfying this requirement and results in large regions of the S^2 domain being negative, especially when we do not initialize $\mathbf{L}^{(0)}$ such that $f_s^{\mathbf{L}^{(0)}}$ is non-negative everywhere.

Having access to intrinsically decomposed shading samples from Intrinsic Dataset allows us to also define a very similar optimization objective, foregoing the reflectance pass, and optimizing directly on shading alone:

$$\hat{\mathbf{L}} = \arg\min_{\mathbf{L}} \quad \frac{1}{J} \sum_j (\Psi_{\text{SH}}(\mathbf{N}_j, \mathbf{L}) - \mathbf{S}_j)^2 \tag{4.4}$$

This should in concept carry a stronger signal, since we avoid a multiplication with potentially low reflectance values. Training with a loss-on-shading is part of our ablation studies in section 6.3.4.

We opt to apply first-order gradient optimization here for several reasons. First off, the photometric objective is non-convex and exhibits many local-minima. We would like the method not to be limited by the high-dimensionality of the chosen parameterization, since one of the most straight-forward extensions to SH lighting is the addition of local, spatially anchored, SH kernels. Depending on how fine the spatial anchoring is, this would lead to the optimization parameter space becoming high-dimensional. Finally, as before, we would like our method to be easily incorporable into NeRF-like pipelines such as [37, 36], which already operate in the same setting.

4.3.2 Normalization and the Non-negativity Constraint

One can argue that when it comes to image or shading data, the information is conveyed on an interval scale – ratios between values are what carry it. This would mean that as long as we can scale and shift the data to a display domain (e.g., $[0, 255] \cap \mathbb{Z}$ for 8-bit images) the untransformed representation can be considered valid. Applying this to our SH representation $\mathbf{L}$, we could say that the hard constraint in eq. (4.3) is superfluous – regardless of what the range of $f_s^{(\mathbf{L})}$ is, we would be able to apply some transformation T before rendering an image such that $T(f_s^{(\mathbf{L})}) \mapsto [0, 1]$. The issue comes with finding an appropriate T.

One common method is to normalize the output of the SH evaluation by the bounds of the data type used to store it. For example, this would map a value stored as an 8-bit unsigned integer with a maximum value of 255 to $[0, 1]$. However, our optimization setting is well-defined only on real values, and the values of

samples in the Intrinsic Dataset are intended to be manipulated within their canonical (unit or [-1,1]) ranges. We can then choose some arbitrary real value as a threshold, and use it for normalization, clipping any over- and underflow. But the choice of the threshold along with clipping may stunt how expressive our SH representation could be. Finally, we could scale each optimization batch by the maximum SH evaluation, This would always result in the maximum shading/lighting value present per batch, which is not realistic for the setting. The drawbacks of these normalization methods leave us with the desire to find $\mathbf{L}$ which would give lighting and shading values also in the canonical range.

For this, [18, 23] propose non-negativity constraint $\mathcal{L}_{\text{neg}}$ on the lighting, since we need not compromise on the choice of the scaling transformation, and use $\hat{\mathbf{L}}$ as is. This constraint quadratically penalizes negativity of K uniform samples over S^2:

$$\mathcal{L}_{\text{neg}} = \sum_k \left(\min\{0,\ \Psi_{\text{SH}}\left(\mathbf{n}_k, \mathbf{L}\right)\} \right)^2 \qquad \text{s.t. } \mathbf{n}_k \sim U\left(S^2\right) \tag{4.5}$$

This leaves us with a full unconstrained optimization formulation which we can solve by gradient descent:

$$\hat{\mathbf{L}} = \arg\min_{\mathbf{L}} \quad \frac{1}{J} \sum_j \mathcal{L}_{\text{PM}}(\mathbf{L} \,|\, \mathbf{R}_j, \mathbf{N}_j, \mathbf{I}_j) + \lambda_{\text{neg}} \cdot \mathcal{L}_{\text{neg}}(\mathbf{L}) \tag{4.6}$$

In our experiments, we see a significant reduction in both negative evaluations and their magnitudes over S^2 (see chapter 6). Pushing the optimization into finding $\hat{\mathbf{L}}$ which maximally maps $f_s^{\hat{\mathbf{L}}}$ to $[0, 1]$ should in essence preserve the expressiveness of the representation into maximally capturing the lighting conditions of the scene.

4.3.3 Implementation Details of Direct Global SH Optimization

The iterative gradient-based optimization is a reworked version of our PyTorch [17] framework implementation used for LightMLP and LightSH. The SH lighting is directly optimized over 3 epochs on the Chair scene and over 5 epochs on the Ficus scene. We use RMSProp [14] which adapts the learning rate via a moving average per-parameter; however, we set the base learning rate to 2.5×10^{-5}. We initialize the SH coefficients to zero $\mathbf{L}^{(0)} = \mathbf{0}$. The non-negativity constraint strength is set to $\lambda_{\text{neg}} = 2$. Each batch consists of 1024 occupied pixels over which we compute the average loss eq. (4.2).In all the presented figures and visualizations, regardless of if we achieve to retrieve $\mathbf{L}$ s.t. $f_s^{\mathbf{L}} \in [0, 1]$, we clip the raw output image data to that range.

Chapter 5

Experiments

In this chapter, we present the experiments we performed to evaluate our methods and their constituents. Through these, we arrive at the final method descriptions found in chapter 4. We carry out parameter sweeps of LightMLP and LightSH to understand the models' mapping capacities and viability in light estimation. We gauge the effect of initialization of SH coefficients $\mathbf{L}$ on our direct optimization approach, evaluate the effect of the non-negative constraint $\mathcal{L}_{\text{neg}}$, and explore whether a direct loss on shading would bring a clearer loss signal and therefore better performance. We present and discuss the results of these experiments in chapter 6.

5.1 Experiments on LightMLP and LightSH

To ascertain the model capacities for supervised-LightMLP, photometric-LightMLP, and LightSH we carry out model hyperparameter sweeps. The sweeps govern model architecture via the layer size and the number of layers, as well as the training procedure by the choice of optimizer, learning rate, dropout rate, and batch size. In the case of LightMLP, we also sweep the strengths of the unitarity constraint. Specifically, with the supervised-LightMLP, the choice of loss function. Note that the layer size hyperparameter dictates the size of each hidden layer of the MLP but the last. The final hidden layer is set to have half the size of the ones preceding it.

To be more specific, we choose the optimizer between Adam and Stochastic Gradient Descent (with a momentum of 0.9). The loss function for LightMLP supervised on GT incident light directions ω_l^* chosen between cosine similarity and mean squared error on the predicted ω_l and ω_l^*. The unitarity constraint strength λ_u is excluded from the hyperparameter sweep of LightSH since the unitarity constraint is not used in that model. For reference and exact description of possible values of all hyperparameters, please refer to the search-spaces and the distributions shown in table 5.1.

Each hyperparameter sweep consists of 10 independently sampled runs of 3 epochs over the full Intrinsic Dataset. We gauge each run's performance by its validation split loss after the final epoch, and by visual inspection of the output. For LightMLP, we augment the Intrinsic Dataset by rendering OLAT samples (described in section 3.4) for each annotated light position in the scene.

The main purpose of the sweeps was to test the general viability of LightMLP and LightSH, and to gauge the utility in expanding them. Nonetheless, we do have some preconceptions about the performance of LightMLP vs. LightSH and on the effect of some hyperparameter choices.

Out of the two LightMLP variations, we expect that photometric-LightMLP would have less of a chance to overfit on an average light incident direction.

Table 5.1: Hyperparameter sweep search-space for LightMLP and LightSH. For each run in the sweep, we choose the value of the hyperparameters according to the listed distribution. QLog$U$ is the quantized log-uniform distribution: it returns $q\lfloor X/q \rceil$ s.t. $\log X \sim U[\log \min, \log \max]$, where $q = 8$.
**Only for LightMLP. **Only for Supervised-LightMLP.*
****Supervised-LightMLP sweep excluded 64 as a possible value for the layer size.*

Hyperparameter	**Distribution**
Loss Function**	$U(\{$'Cosine Similarity', 'MSE'$\})$
Optimizer	$U(\{$'Adam', 'SGD'$\})$
Learning Rate	$U[0.0001, 0.1]$
Number of Layers	$U(\{3, 6, 10\})$
Layer Size	$U(\{64, 128, 256, 512\})$***
Dropout Rate	$U(\{0.15, 0.2, 0.25, 0.3, 0.4\})$
Batch Size	$\text{QLog}_8 U[32, 256]$
Unitarity Strength* λ_u	$U[0, 2]$

Note that ω_l – depending on the proximity of the light l to the object – would have a small and slowly varying range over its surface. This is certainly the case for the scenes in the Intrinsic Dataset.

Furthermore, it is possible that due to the disconnectedness of individual predictions that the output space of LightMLP would not be smooth, such that the average loss is small, however the output exhibits splotchiness.

With runs of the supervised-LightMLP we expect that the cosine similarity loss would give a better training signal than a less-forgiving MSE. The inclusion of the unitarity constraint should lead to well-behaved vector norms. After all, it is the direction of the predicted vector that we value most, since a non-unit vector output can be normalized before use.

LightSH has promise in that spherical harmonics are always smooth functions. Because of this, if the model finds a smooth output landscape w.r.t. to the inputs, it would be able to model multi-source local lighting as the predicted SH representation varies across the object surface. We, however, note, that LightSH does not have any extra mechanisms that would enforce or encourage smoothness of output from neighbouring inputs. That it will predict smooth SH lighting is a hope. So we should not expect a perfectly smooth output, but still, one that performs better than LightMLP.

In general, the effect of model size should be seen in the results; larger MLPs would have the capacity to discriminate over small changes in the target direction vectors with LightMLP. However, there is a possibility that a larger model has too much capacity for the relatively low-dimensional input we give it. Perhaps, the inclusion of a spatial input would help in this regard.

5.2 Experiments on Direct SH Optimization

5.2.1 Initialization of SH coefficients L

For any optimization problem, the initialization of its optimization parameters should be considered carefully. The initial point may steer the optimization to a local minimum or divergence when the optimization objective is non-convex. When initializing weights of a neural network, it is common to initialize them randomly.

In our optimization formulation (eq. (4.6)) we note that since we require the predicted lighting $\hat{\mathbf{L}}$ to induce shading $\Psi_{\text{SH}}(\cdot, \hat{\mathbf{L}}) \in [0, 1]$ it should follow that $-\epsilon \leq \hat{\mathbf{L}}_i \leq \epsilon$ for some small positive value of ϵ; the SH coefficients should be close to zero.

We are curious about the effect of initialization of $\mathbf{L}$ on the course of optimization. Would a standard normally distributed initialization $\mathbf{L}^{(0)} \leftarrow \mathcal{N}(0,1)$ frequently result in non-optimal solutions? Would it instead help the optimization steer away from such? On the other hand, would an initialization of non-lighting, that is, zero lighting $\mathbf{L}^{(0)} \leftarrow \mathbf{0}$ be a good starting point? We consider it a viable candidate since it is the simplest physically consistent configuration of $\mathbf{L}$.

To find out, we construct a simple comparison between these two. We carry out 10 runs of the optimization per initialization method. Each random initialization is sampled independently of each other from $\mathcal{N}(0,1)$. All runs are optimized with RMSProp with a learning rate of 2.5×10^{-5}, with all other optimization settings occur to those described in section 4.3.3. These runs train over the full-render passes of Intrinsic Dataset's chair scene, and are run for several epochs until no significant change to $\mathbf{L}$ occurs for one epoch. We record the evolution of $\mathbf{L}$ during each run for later analysis.

We expect that with enough time to converge, all runs will come to a consensus for the approximate value of $\mathbf{L}$. However, we are unsure how long convergence would take in either case on average. If our hypothesis that in most cases the magnitude of $\hat{\mathbf{L}}$ should be low is correct, a zero initialization would conceivably take less optimization steps to reach that point.

5.2.2 Efficacy of the Non-negativity Constraint

Achieving non-negative lighting is important since it would be physically plausible. This would avoid 'wasted' model capacity to express negativity, and avoid clipping, normalizing, or scaling the shading to form an image. For our estimates of $\mathbf{L}$ to be physically plausible, we need to set λ_{neg} to an appropriate value.

We conduct a simple search to do this. For each initialization method from section 5.2.1, we carry out runs of our optimization while gradually increasing the non-negativity weight λ_{neg} from 0 to 3 with increments of 0.5. This gives us a total of 14 runs. When we initialize $\mathbf{L}^{(0)} \leftarrow \mathcal{N}(0,1)$, we use a learning rate of 1.6×10^{-4} such that all runs may sufficiently converge within 3 epochs in tandem with the zero-initialization.

Note that with $\mathbf{L}^{(0)} \leftarrow \mathcal{N}(0,1)$ the initial shading has no guarantee of being physically plausible, and may very well be purely negative, or purely exceeding one over S^2. Therefore, we also have no guarantee that the representation will converge to be within [0,1].

Our hypothesis is then that in runs with $\lambda_{\text{neg}} = 0$, negative portions over S^2 will not fully go away, and may even remain substantial in portions where little incident lighting is present. Higher values of λ_{rm} should eliminate the issue, but setting it too high might bring imbalance to the optimization – where non-negativity would be enforced at the expense of modelling accurate lighting elsewhere in the scene.

5.2.3 Using a Direct Loss-on-shading

We premise our method on the existence of an intrinsically decomposed neural field of geometry, radiance, shading, and reflectance. So, one could argue that instead of relying on the photometric reconstruction from both shading and reflectance, we ought to use shading directly to approximate $\mathbf{L}$. After all, does shading not directly indicate lighting, since the geometry is assumed to be known? Why complicate things?

We, too, had this thought, and so propose the following experiment. As we show in fig. 4.3, we can use the track shown in yellow to derive a pseudo-photometric loss on just the shading. We would compare the predicted shading $\Psi_{\text{SH}}(\cdot, \mathbf{L})$ to the shading pass $\mathbf{S}$ from the Intrinsic Dataset. Pseudo-photometric, since it acts on greyscale values of shading.

To evaluate if a direct loss-on-shading would be beneficial to our method, we include an ablation study on in section 6.3.4.

Chapter 6

Results and Discussion

6.1 Error Metrics

Throughout this chapter, we will be using a collection of error metrics to evaluate our experiments and results. These metrics (and their implementation details) are as follows:

- MSE is the mean squared error between corresponding render or shading image elements. In our case, this is always a mean over the number of elements rather than the number of pixels (e.g., a three channel RGB image of size (W, H) would have $W \cdot H \cdot 3$ elements in it). When applying MSE to rendered colour pixels, we get the photometric loss we have been using for training (see section 4.3).

- LMSE is the local version of MSE and is supposed to measure a more granular metric on square patches of the predicted and ground truth images. The intent is to capture how well each local region is reconstructed. A sliding window is used to extract corresponding patches of each image, which are compared via MSE. The result of the metric is the average of the MSEs of the patches. By a recommendation in [9] the size of the patches in set to 10% of the longest image side (40 pixels in our case). We slide the window by half of its width in each dimension each iteration. [6]

- SIL2 or scale-invariant L2 loss, also known as scale-invariant MSE, is a modified MSE metric recommended by [6]. Since shading lies on an interval scale, many authors recommend using a scale-invariant metric to mitigate penalties of wrongly predicting absolute scale while having the correct intervals, or conversely, preventing good MSE evaluations when just the mean of the shading is predicted well. We use the definition from [34, 9, 8]:

$$SIL2(I, I^*) = \frac{1}{N} \sum \left(\ln I - \ln I^* \right)^2 - \frac{\lambda}{N^2} \left(\sum \ln I - \ln I^* \right)^2 \tag{6.1}$$

  we use $\lambda = 0.5$ where N is the number of pixels. Before evaluation, we clip the images between 1×10^{-5} and 1 to prevent NaNs. Since all of our methods are trained on the usual MSE, this metric is presented for reference only. LSIL2 is the local version of this metric, identical to LMSE, except that SIL2 is used instead of MSE on each patch.

- PSNR or peak signal-to-noise ratio is a measure of distortion in an image compared to a ground truth and is measured on the logarithmic scale. Out of all the used metrics, this is the only one for which higher values indicate better reconstruction. PSNR is defined through the value of the MSE of the input images as $\text{PSNR} = 10 \cdot \log_{10}(\frac{\text{MAX}_I}{\text{MSE}}) = -10 \cdot \log_{10}(\text{MSE})$ where $\text{MAX}_I = 1$ is the maximum possible pixel value in our images. Though we should note that (like the numeric-quantitative metrics above) PSNR does not always correlate well with the human perception of image quality or similarity.

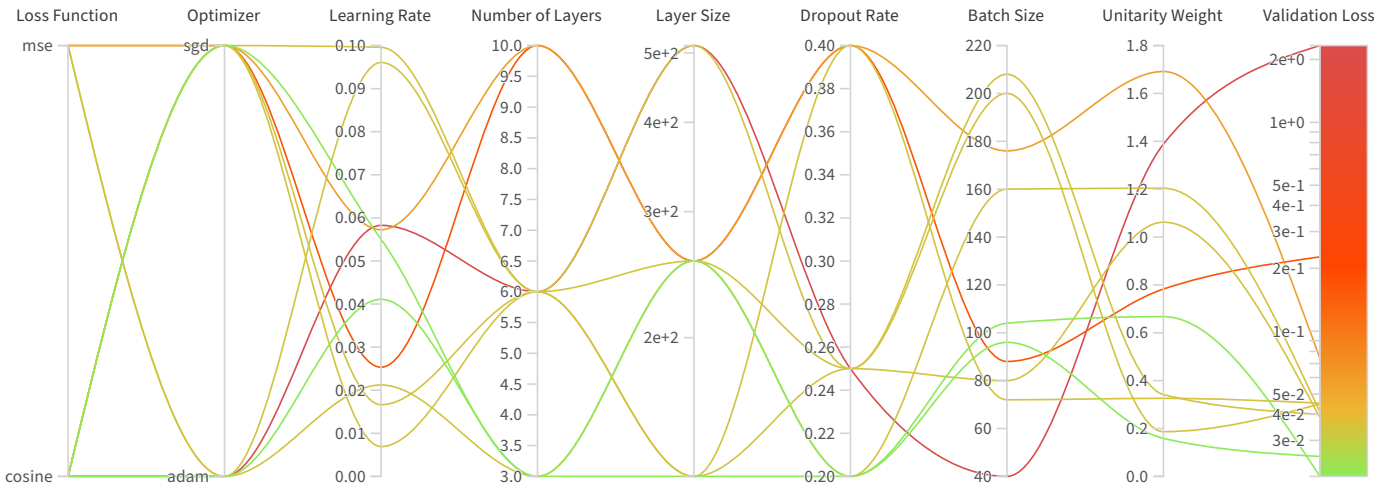

Figure 6.1: Parallel Coordinates plot for hyperparameter sweep of supervised-LightMLP. Validation loss shown on a logarithmic scale for clarity.

- DSSIM is the inverse version of the structural similarity index (SSIM) [5] metric, where DSSIM $= \frac{1-\text{SSIM}}{2}$. While MSE-based metrics give comparisons on the numerical values of our predictions, DSSIM quantifies the human perception dissimilarity between two images and, like LMSE and LSIL2, operates on patches of these. It captures aspects of the that a simple mean error metric does not, like differences in contrast, luminance, and structure in the images.

6.2 Results of LightMLP and LightSH

We show the parallel coordinate plots of the hyperparameter sweeps of our MLP-based model in fig. 6.1, fig. 6.2, and fig. 6.3. We show the visual output of the best configurations of each model in fig. 6.4, fig. 6.5 , and fig. 6.3 respectively. From the parallel coordinate plots, we cannot discern any single combination of parameters which would immediately stand out as the best performing. This, except for the sweep for LightSH, where there is a clear outlier, but even then, we have not a solid stance on the reasoning. It appears that the choice of smaller layer sizes leads to better overall validation loss. We do want to point out that the supervised model shows better performance using the cosine loss.

Nonetheless, looking at the visual output of supervised-LightMLP, we see that mostly the predicted light direction corresponds to GT. But we also see that the model predicts a large artefact across the surface of the chair. This would not be ideal if we want to use these directions for light source estimation later. In any case, this is a supervised model, which we cannot use further due to the nature of light source estimation in the first place.

Photometric-LightMLP shows good visual results, however these are marred with splotchiness. This can be easily seen in the predicted light directions, which are non-spatially coherent, to the scene or each other's neighbours.

Finally, LightSH shows the best performance. We see that their shading reconstruction is generally smooth, but light LightMLP lacks local smoothness. Combining this model for local SH lighting with the global SH optimization could be considered future work.

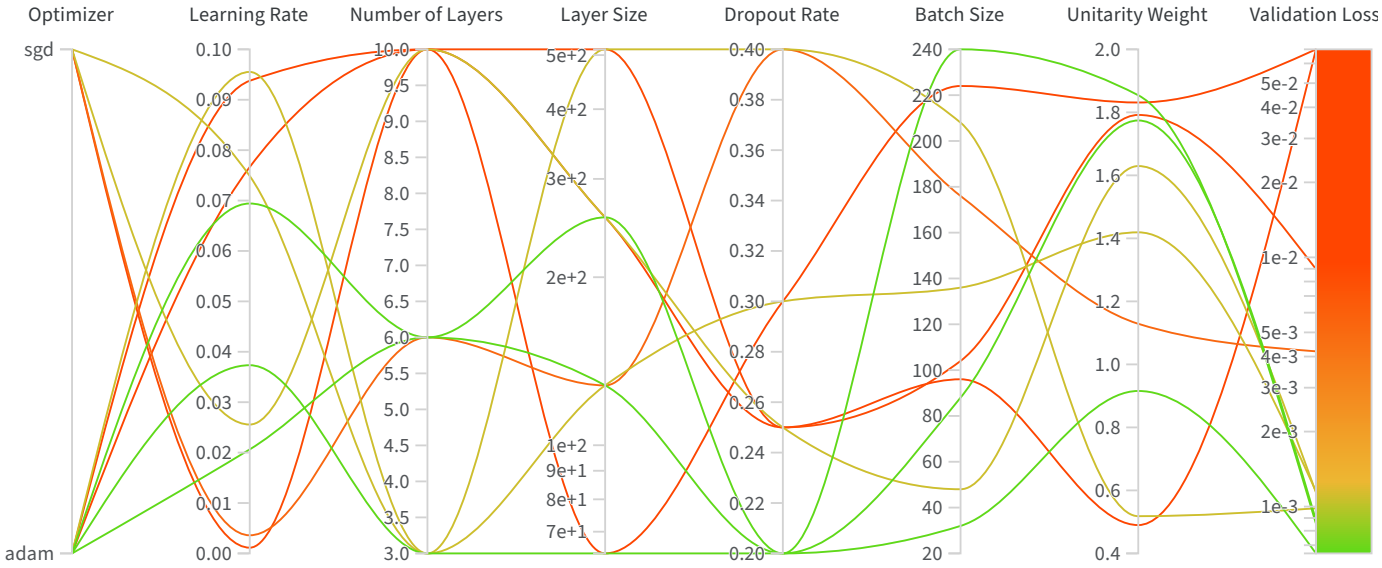

Figure 6.2: Parallel Coordinates plot for hyperparameter sweep of photometric-LightMLP. Note that validation loss is shown on a logarithmic scale for clarity.

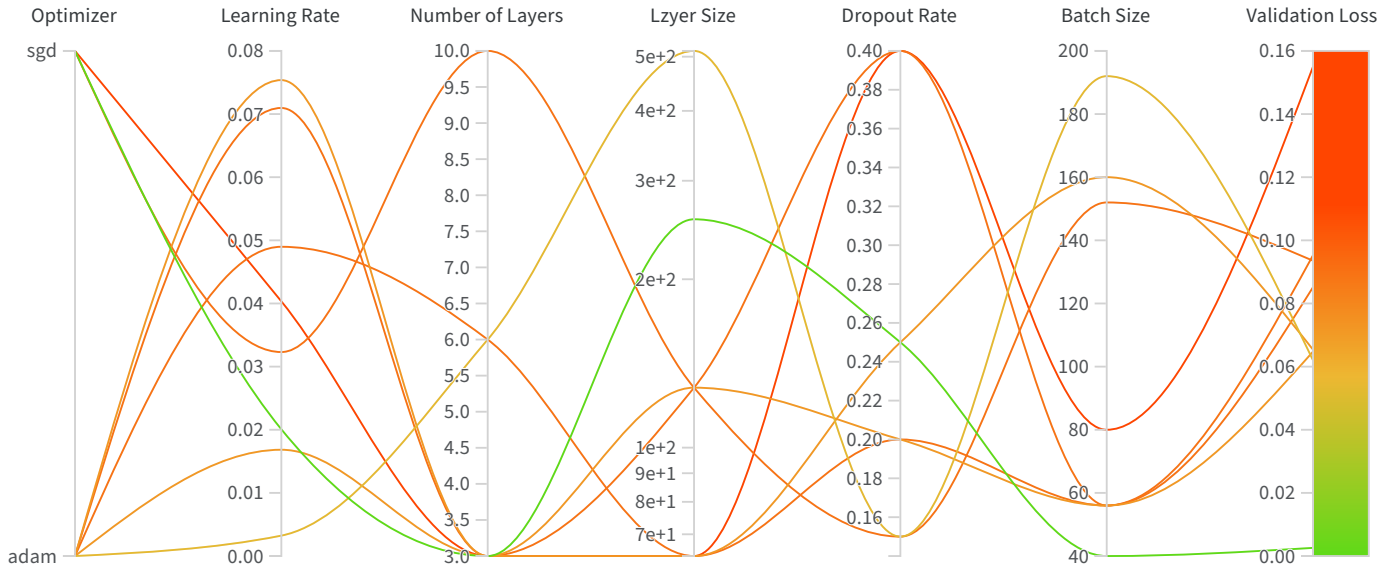

Figure 6.3: Parallel Coordinates plot for hyperparameter sweep of LightSH. Note that we do not show two failed runs out of the 10 we carried out.

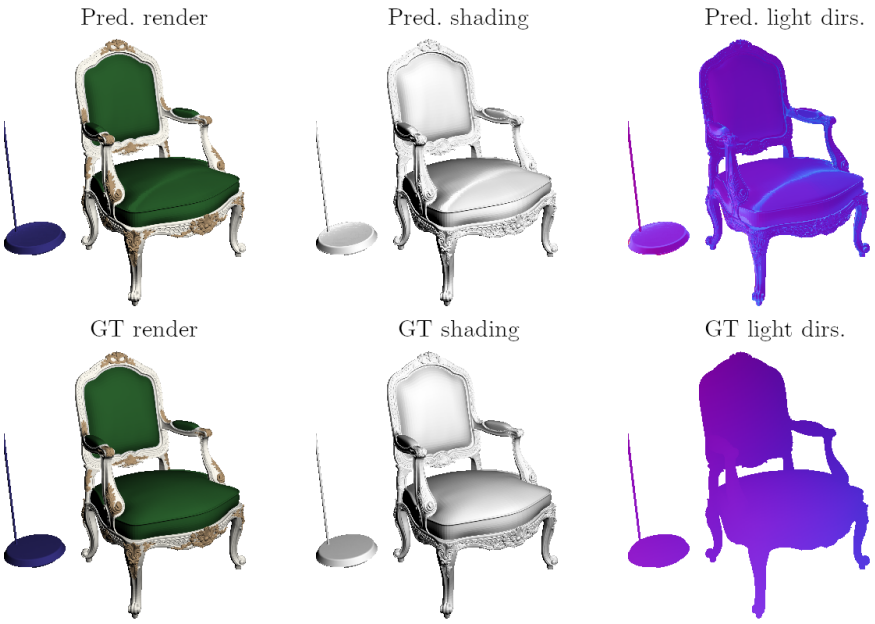

Figure 6.4: Visualized render, shading, emerging from predicted light directions of supervised-LightMLP.

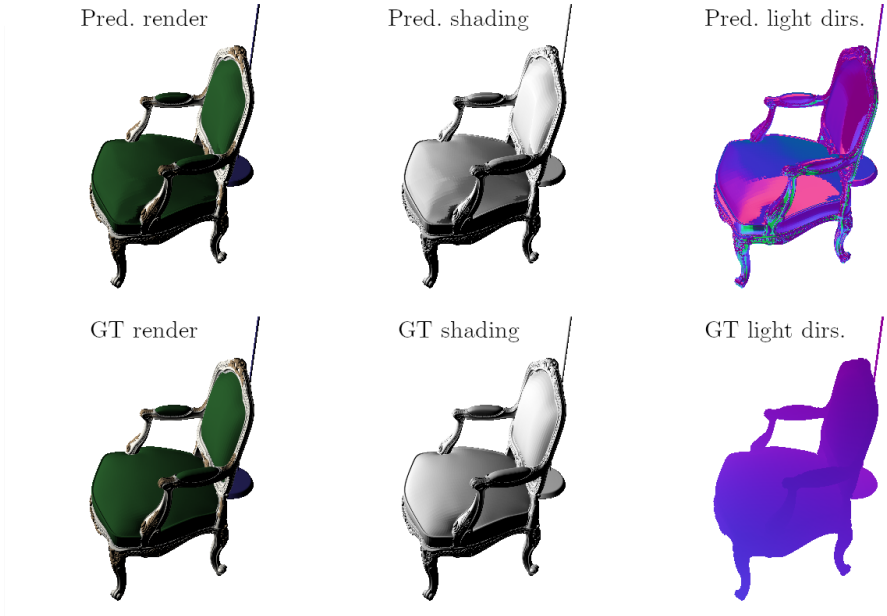

Figure 6.5: Visualized render, shading, emerging from predicted light directions of photometric-LightMLP.

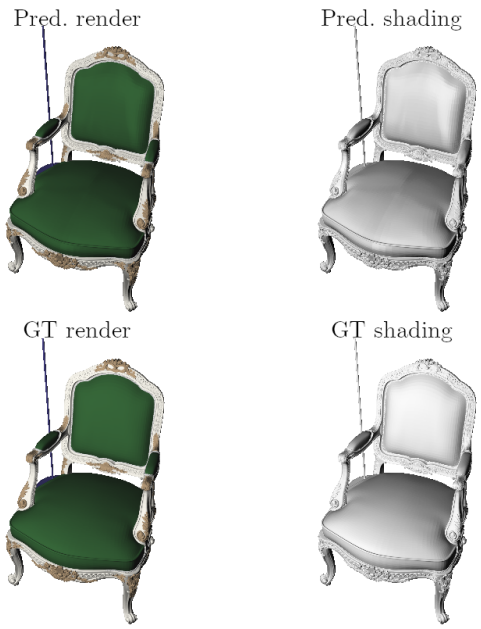

Figure 6.6: Visualized render from predicted shading of LightSH.

6.3 Results of Direct SH Optimization

6.3.1 Initialization of SH Coefficients

The results of the optimization initialization experiments are shown in fig. 6.7 and fig. 6.8. In the former, we show the evolution of the SH coefficients of a single representative randomly-sampled-initialization run. The shaded regions indicate the span of the values of each coefficient during the optimization. The optimization consistently took 20 epochs to settle on the final coefficient values. In the latter, we show the average evolution of the SH coefficients for runs which have all been initialized at the origin. We still display the span of each coefficient; however, it is indiscernible due to the low variation between runs. Runs initialized at zero consistently converge to the same values as the sampled runs but take less time to do so.

We think there are two reasons for this. One, on average, each coefficient must be updated for fewer steps to reach its optimal value if we start at a point already close to it – in this case that is the origin. Second, a random initialization often results in drastic changes in all coefficient values in the beginning steps of the optimization. This leads to wider spread out and therefore longer convergence.

6.3.2 Effect of the Non-negativity Constraint

We show the results of our experiments on the effect of the non-negative constraint $\mathcal{L}_{\text{neg}}$ in fig. 6.9 though fig. 6.12.

If we do not apply $\mathcal{L}_{\text{neg}}$, by setting λ_{neg} to zero, we see the following. With randomly initialized runs, large portions of lighting evaluate to large negative magnitudes. Figure 6.10 shows that the whole lower half of the lighting sphere is negative. We had to truncate the histogram in fig. 6.9 to be comparable to the rest. Note that in the raw data, the evaluations ranged to a minimum of -3.9.

When we initialize the SH coefficients to zero, yet don't apply $\mathcal{L}_{\text{neg}}$, it appears that from the start, the optimization has no incentive to converge to a negative state. Figure 6.13 shows less than 500 evaluations below zero in this case, and the negative solid angle in fig. 6.14 is much smaller than in fig. 6.10. Going further, fig. 6.15 shows a decrease in the magnitudes of only those evaluations already below zero, yet the

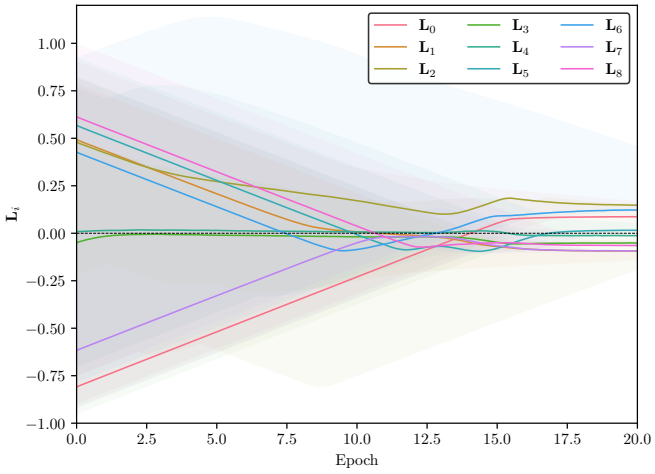

Figure 6.7: Evolution of SH coefficients $\mathbf{L}$ for a randomly initialized optimization. We show the span of each $\mathbf{L}_i$ with the shaded regions. Note that this optimization took 20 epochs to complete.

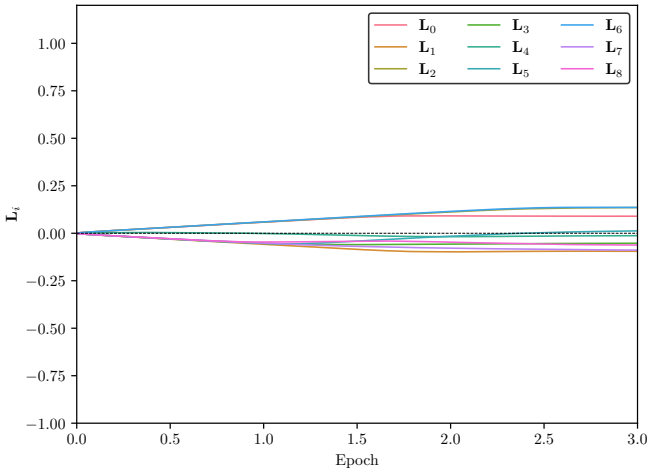

Figure 6.8: Average evolution of SH coefficients $\mathbf{L}$ over 10 zero-initialized runs. Note that this optimization only takes 3 epochs to complete.

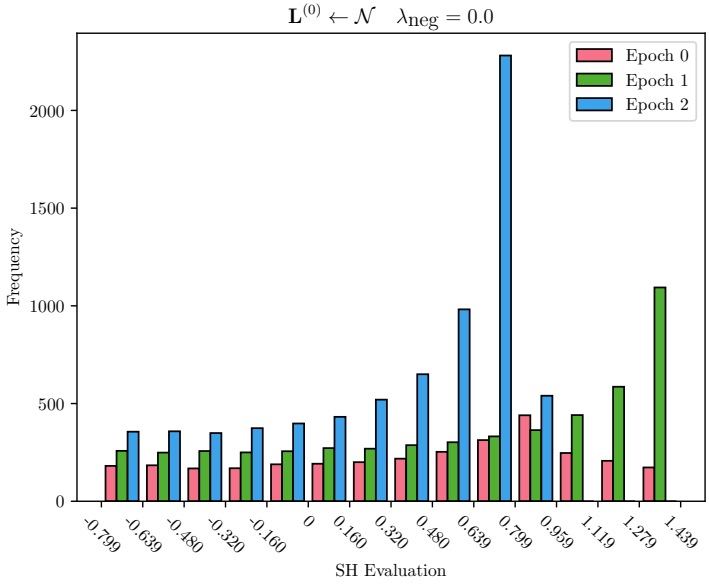

Figure 6.9: Histogram of SH evaluations, $\mathbf{L}^{(0)} \leftarrow \mathcal{N}$, $\lambda_{\text{neg}} = 0$. We truncate the x-axis to show the first 5 bins which fall below zero. In actuality, these evaluations ranged to a minimum of -3.9.

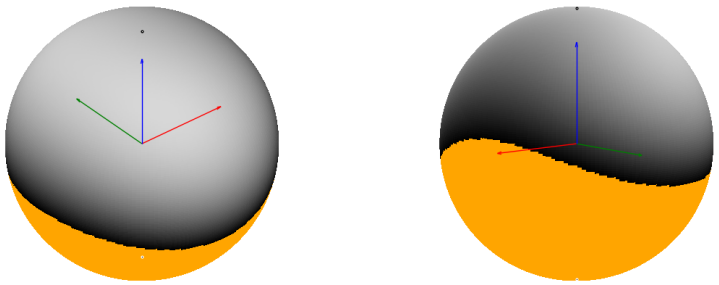

Figure 6.10: Visualized SH lighting on a sphere, $\mathbf{L}^{(0)} \leftarrow \mathcal{N}$, $\lambda_{\text{neg}} = 0$. Orange regions mark negative evaluations.

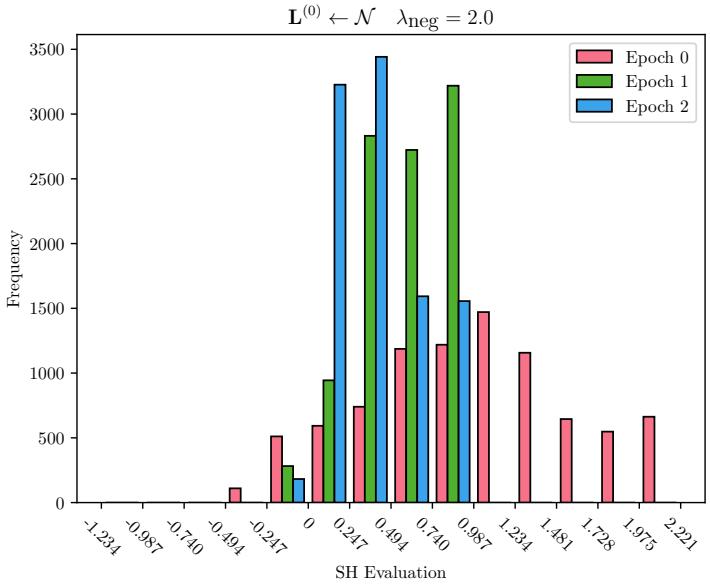

Figure 6.11: Histogram of SH evaluations, $\mathbf{L}^{(0)} \leftarrow \mathcal{N}$, $\lambda_{\text{neg}} = 2$

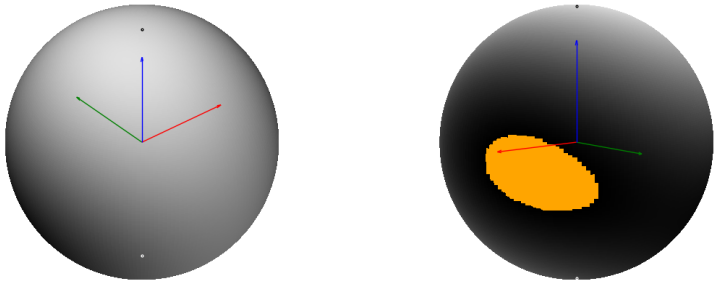

Figure 6.12: Visualized SH lighting on a sphere, $\mathbf{L}^{(0)} \leftarrow \mathcal{N}$, $\lambda_{\text{neg}} = 2$. Orange regions mark negative evaluations.

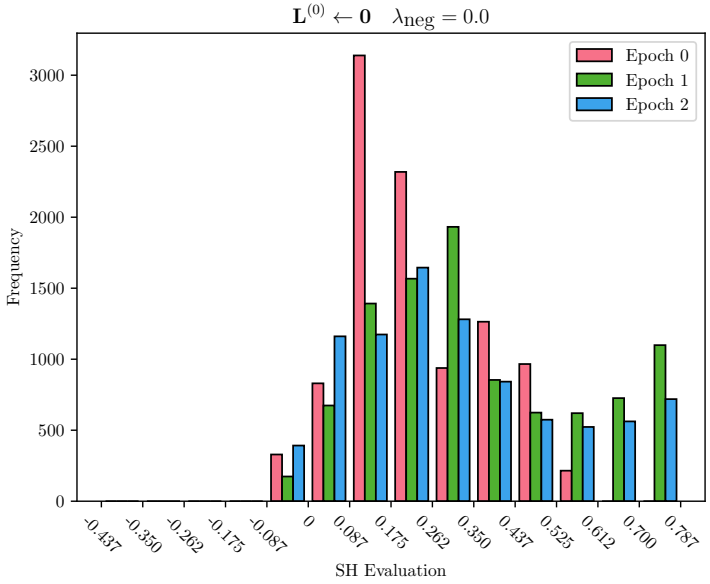

Figure 6.13: Histogram of SH evaluations, $\mathbf{L}^{(0)} \leftarrow \mathbf{0}$, $\lambda_{\text{neg}} = 0$

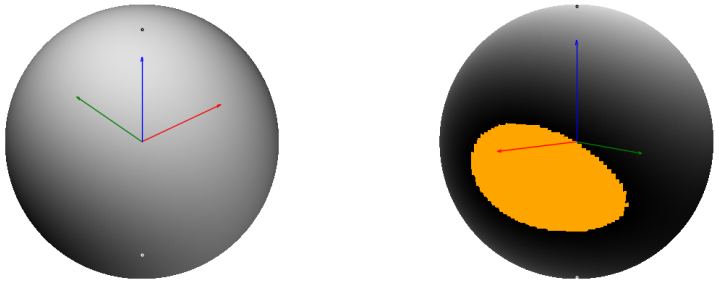

Figure 6.14: Visualized SH lighting on a sphere, $\mathbf{L}^{(0)} \leftarrow \mathbf{0}$, $\lambda_{\text{neg}} = 0$. Orange regions mark negative evaluations.

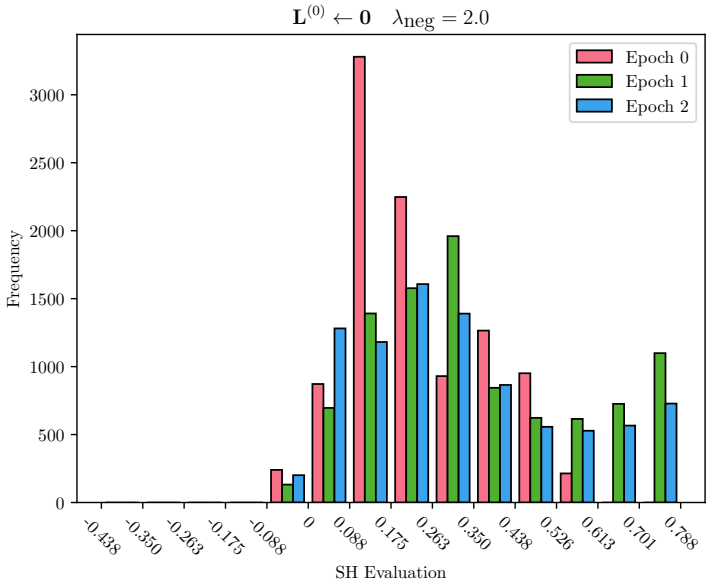

Figure 6.15: Histogram of SH evaluations, $\mathbf{L}^{(0)} \leftarrow \mathbf{0}$, $\lambda_{\text{neg}} = 2$

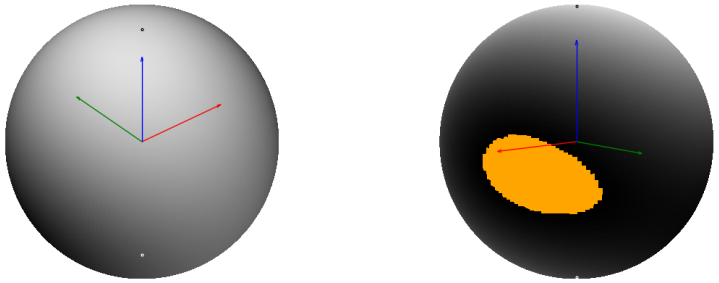

Figure 6.16: Visualized SH lighting on a sphere, $\mathbf{L}^{(0)} \leftarrow \mathbf{0}$, $\lambda_{\text{neg}} = 2$. Orange regions mark negative evaluations.

entire positive part of the histogram is barely changed.

This shows that applying the non-negativity constraints is viable in all modes of our method. It both reduces the solid angle of negative evaluations over S^2 and also decreases the magnitudes of these evaluations such that they may be neglected to clipping. This result further leads us to use the zero-initialization in our method, since it is the more favourable setting w.r.t. plausible lighting as well.

6.3.3 Direct Loss-on-shading

6.3.4 Ablation Studies

Table 6.1: Ablation over the initialization of the SH lighting $\mathbf{L}$, addition of the non-negativity constraint $\mathcal{L}_{\text{neg}}$, and training directly on ground truth shading via $\mathcal{L}_{\text{S}}$. Metrics are evaluated over the `test` split of Intrinsic Dataset using the 'full' colour render, and are averaged over three runs. We show the evaluations of the metrics over the RGB render image pairs, as well as the greyscale shading pairs. Lower is better, except for PSNR.

	Render						Shading					
	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM
$\mathbf{L}^{(0)} = \mathcal{N}$	0.0282	0.00722	15.39	4.15	15.69	0.0694	0.0373	0.00996	9.62	2.20	14.73	0.0475
$+ \mathcal{L}_{\text{neg}}$	0.0263	0.00674	14.74	3.43	15.97	0.0691	0.0329	0.00871	8.90	2.03	15.28	0.0429
$+ \mathcal{L}_{\text{S}}$	0.0276	0.00709	16.52	3.12	15.72	0.0711	0.031	0.00836	9.05	2.09	15.69	0.0427
$\mathbf{L}^{(0)} = \mathbf{0}$	0.026	0.00669	14.10	3.49	16.00	0.0678	0.0329	0.00877	8.87	2.05	15.32	0.0419
$+ \mathcal{L}_{\text{neg}}$	0.0262	0.00671	14.84	3.40	15.98	0.0688	0.0328	0.00875	8.87	2.03	15.34	0.0424
$+ \mathcal{L}_{\text{S}}$	0.0276	0.0071	16.74	3.05	15.72	0.0714	0.0313	0.00846	8.99	2.08	15.65	0.0433

We compile ablations of our direct optimization method in table 6.1. As a baseline, we see that initializing $\mathbf{L}$ at the origin performs better on all metrics, except for the comparison of shading via DSSIM. We

Table 6.2: Results of direct SH optimization on the chair scene.

	Render						Shading					
	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM
chair, diffuse, $\mathcal{L}_{\text{PM}}$	0.0134	0.00343	14.82	3.08	18.99	0.0559	0.0417	0.0111	8.77	2.00	14.00	0.0455
chair, diffuse, $\mathcal{L}_{\text{S}}$	0.0168	0.00433	16.68	2.67	17.90	0.0609	0.0313	0.00846	8.99	2.08	15.65	0.0433
chair, full, $\mathcal{L}_{\text{PM}}$	0.0261	0.00671	14.83	3.40	15.98	0.0688	0.0329	0.00876	8.86	2.03	15.33	0.0425
chair, full, $\mathcal{L}_{\text{S}}$	0.0276	0.0071	16.75	3.05	15.72	0.0714	0.0313	0.00846	9.00	2.08	15.64	0.0434

Table 6.3: Results of direct SH optimization on the ficus scene.

	Render						Shading					
	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM	MSE	LMSE	SIL2	LSIL2	PSNR ↑	DSSIM
ficus, diffuse, $\mathcal{L}_{\text{PM}}$	0.0182	0.00249	1.49	0.242	17.95	0.0394	0.0914	0.0139	6.78	0.923	10.46	0.0397
ficus, diffuse, $\mathcal{L}_{\text{S}}$	0.0207	0.00302	1.79	0.256	17.02	0.0453	0.0579	0.00913	7.11	0.978	13.23	0.0271
ficus, full, $\mathcal{L}_{\text{PM}}$	0.0354	0.00516	1.78	0.299	14.61	0.0527	0.0651	0.0101	6.90	0.946	12.29	0.03
ficus, full, $\mathcal{L}_{\text{S}}$	0.0364	0.0054	1.89	0.298	14.46	0.054	0.0579	0.00914	7.11	0.978	13.22	0.0271

note that the lower values in the render MSE column, when compared to those in the shading MSE column, could likely be explained by the addition of reflectance in the former. Here, the reflectance with is within $[0, 1]$ and hence scales the shading image magnitudes down, in turn causing lower MSE. We provide SIL2 and LSIL2 metrics for reference, and note that none of our methods use these during training/optimizing. We suggest that the lower SIL2 metrics, when comparing shading, result from the view-dependent and other non-Lambertian effects present in the full render passes that we do not store in the reflectance-only passes of the Intrinsic Dataset - shading is therefore more similar by this limitation.

Adding the non-negativity constraint improves our metrics throughout all comparisons when we initialize with randomness. When we initialize with zero, adding $\mathcal{L}_{\text{neg}}$ does not improve the output. This is expected because when initialized to zero does not become substantially more negative even without the constraint.

We observe that adding the loss-on-shading worsens the model. We note that, again, this table shows our evaluations on the 'full' render passes on the Intrinsic Dataset. When we compare the effect of $\mathcal{L}_{\text{S}}$ between evaluations on the diffuse pass and 'full' passes (like we do in table 6.2) we see that the loss-on-shading is better on diffuse scenes. Since we do not expect purely diffuse scenes in the wild, we choose to show our final results using the usual photometric loss.

6.3.5 Results on Test-split of Intrinsic Dataset

We show the final visual results of the SH lighting optimization in fig. 6.17 for the chair scene, and in fig. 6.18 for the ficus scene. The average metrics over the test split of the Intrinsic Dataset are shown in table 6.2 and table 6.3.

We observe the following:

- For both scenes, we can estimate the global lighting cues of the scene.
  - From row 1 of fig. 6.17 shows that the chair is lit from above and the front, while row 3 shows that it is not lit from the back. This is consistent with the two lights present in the scene.
  - The ficus scene, which has 3 lights which light the plant from most direction, is also reconstructed reasonably with the SH

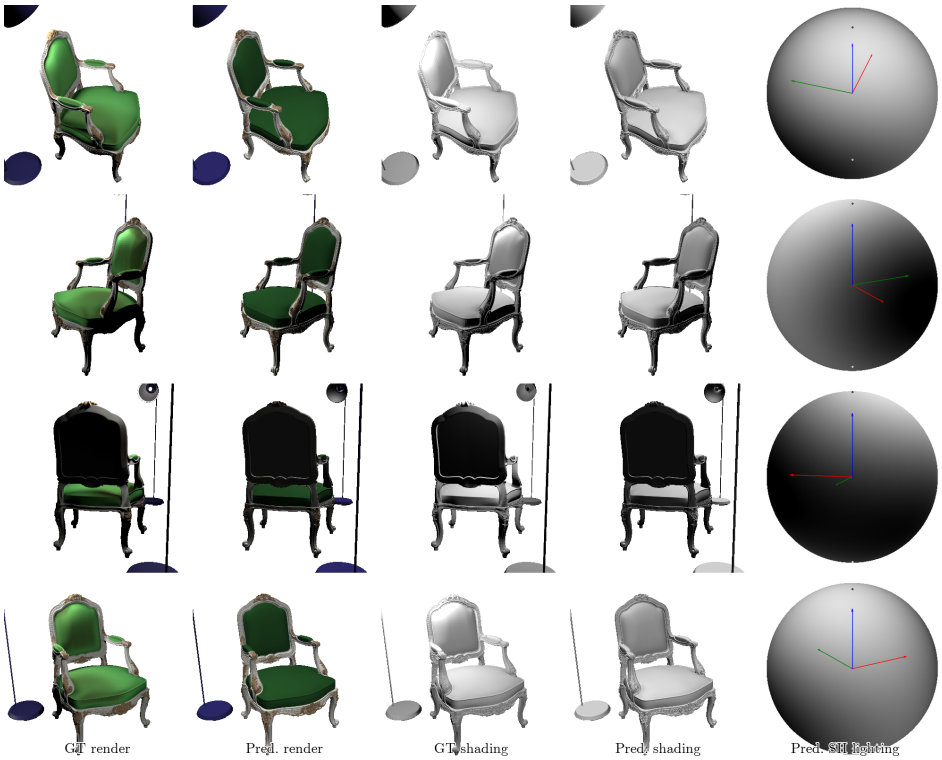

Figure 6.17: Results on the 'full' render pass of the chair scene. Left to right: GT render, predicted render, gt shading, predicted shading, and visualized SH lighting.

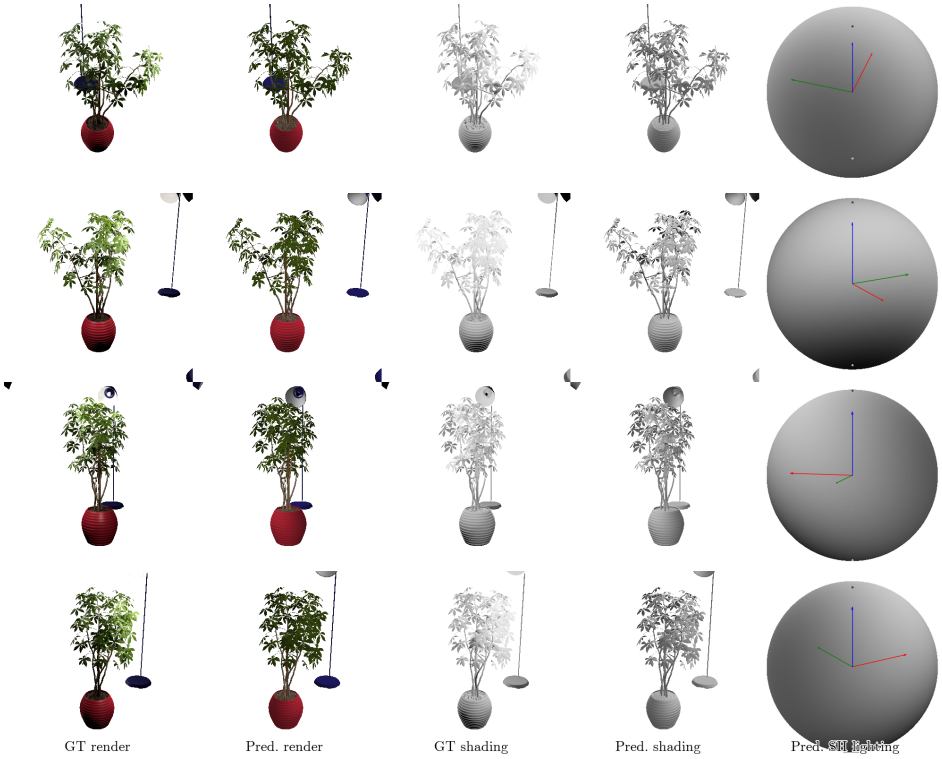

Figure 6.18: Results on the 'full' render pass of the ficus scene. Left to right: GT render, predicted render, gt shading, predicted shading, and visualized SH lighting.

- We note that our estimate of the ficus scene lighting is significantly darker than GT. We think this is due to the capacity of the global SH lighting model.
  - The ficus scene has more complex and varied geometry, with many similarly oriented surfaces (leaves) being lit differently.
  - The ficus scene also has more dispersed lighting.
  - This requires the SH representation to both show bring lighting everywhere, but also shading on the bottoms of the leaves, which it struggles to do simultaneously.

Chapter 7

Conclusion and Outlook

In this work, our main aim was to propose a method of light source estimation within a NeRF-like setting. To do so, we leveraged recently proposed NeRF flavours capable of intrinsic image decomposition. This enabled us to potentially use a field of disentangled reflectance, shading, and geometry to infer lighting conditions.

We found that a direct neural mapping from intrinsic components to lighting is ineffective when using simple MLPs. In our work, these are not spatially anchored to the scene, and lack the constraints needed to ensure smooth inferred lighting. In our exploration of these predictive models, we did not find model hyperparameters which differentiate one model architecture from another. This is not to imply that all considered models are non-viable. LightSH, even without the extra machinery of spatial or neighbourhood constraints, performed well enough to infer an acceptable overall lighting reconstruction.

We pivot to the exploration of a direction optimization approach of SH lighting. We found that we can retrieve a coarse global representation of lighting which is consistent with the scene's local lights. In our experiments, we found that initializing the SH representation to no lighting gives benefits to the stability and speed of the optimization. We note that although the inclusion of a non-negativity constraint did reduce the magnitudes of non-physically plausible lighting evaluation, these were no substantial enough in the first place when initializing at the origin. Rather curiously, we note that a direct loss-on-shading performs better on purely diffuse scenes over the usual photometric loss.

7.1 Future Work

To avoid scope creep, we made conscious decisions of what to focus on in this thesis. The decision to work with our Intrinsic Dataset instead of the (at that time unpublished) intrinsically decomposed fields. Hence, some avenues that we could have researched were not. We propose that future work could include the following:

- Improvement of the Intrinsic Dataset in included view-dependant reflectance and decompositions of residual terms like specularities.
- Inclusion and anchoring of a local SH light representation in the scene. The addition of SVSH through, for example, a variation of LightSH with the global representation as a basis. This would increase the capacity of types of lighting we can estimate.
- Replacing the input from Intrinsic Dataset with samples from an intrinsically decomposed field. This would require working the implementation of our work to act on ray queries instead of pixel values in

the Intrinsic Dataset.

- Use of the lighting estimate in tandem with neural field training, its join optimization. Lighting cues could be used to better understand shading and vice versa.

Bibliography

[1] Harry Barrow et al. 'Recovering Intrinsic Scene Characteristics'. In: *Comput. vis. syst* 2.3-26 (1978), p. 2.

[2] Steven A. Shafer. 'Using Color to Separate Reflection Components'. In: *Color Research & Application* 10.4 (Dec. 1985), pp. 210–218. ISSN: 0361-2317, 1520-6378. DOI: 10.1002/col.5080100409. URL: https://onlinelibrary.wiley.com/doi/10.1002/col.5080100409 (visited on 24/11/2023).

[3] Michael Landy and J. Anthony Movshon. 'The Plenoptic Function and the Elements of Early Vision'. In: *Computational Models of Visual Processing*. 1991, pp. 3–20.

[4] R. Basri and D.W. Jacobs. 'Lambertian Reflectance and Linear Subspaces'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.2 (Feb. 2003), pp. 218–233. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1177153. URL: http://ieeexplore.ieee.org/document/1177153/ (visited on 01/11/2023).

[5] Zhou Wang et al. 'Image Quality Assessment: From Error Visibility to Structural Similarity'. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[6] Roger Grosse et al. 'Ground-Truth Dataset and Baseline Evaluations for Intrinsic Image Algorithms'. In: *International Conference on Computer Vision*. 2009, pp. 2335–2342. DOI: http://dx.doi.org/10.1109/ICCV.2009.5459428.

[7] Jorge Lopez-Moreno et al. 'Multiple Light Source Estimation in a Single Image: Multiple Light Source Estimation in a Single Image'. In: *Computer Graphics Forum* 32.8 (Dec. 2013), pp. 170–182. ISSN: 01677055. DOI: 10.1111/cgf.12195. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12195 (visited on 03/08/2023).

[8] David Eigen, Christian Puhrsch and Rob Fergus. 'Depth map prediction from a single image using a multi-scale deep network'. In: *Advances in Neural Information Processing Systems*. Vol. 3. Neural information processing systems foundation, 2014, pp. 2366–2374.

[9] Takuya Narihira, Michael Maire and Stella X. Yu. 'Direct Intrinsics: Learning Albedo-Shading Decomposition by Convolutional Regression'. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2992–2992. DOI: 10.1109/ICCV.2015.342.

[10] cenksns. *Ikea Hektar Floor Lamp 3D Model*. TurboSquid, 18th Jan. 2016. URL: https://www.turbosquid.com/3d-models/free-ikea-hektar-floor-lamp-3d-model/996693 (visited on 26/11/2023).

[11] Thomas Whelan et al. 'ElasticFusion: Real-time Dense SLAM and Light Source Estimation'. In: *The International Journal of Robotics Research* 35.14 (Dec. 2016), pp. 1697–1716. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364916669237. URL: http://journals.sagepub.com/doi/10.1177/0278364916669237 (visited on 04/08/2023).

[12] Robert Maier et al. 'Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017 IEEE International Conference on Computer Vision (ICCV). Venice: IEEE, Oct. 2017, pp. 3133–3141. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.338. URL: https://ieeexplore.ieee.org/document/8237600/ (visited on 01/06/2023).

[13] Peter-Pike Sloan. 'Deringing Spherical Harmonics'. In: *SIGGRAPH Asia 2017 Technical Briefs*. SA '17. New York, NY, USA: Association for Computing Machinery, 2017. ISBN: 978-1-4503-5406-6. DOI: 10.1145/3145749.3149438. URL: https://doi.org/10.1145/3145749.3149438.

[14] Geoffrey Hinton. *Neural Networks for Machine Learning: Overview of Mini-batch Gradient Descent*. Feb. 2018. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (visited on 07/11/2023).

[15] Marc-André Gardner et al. *Deep Parametric Indoor Lighting Estimation*. 19th Oct. 2019. arXiv: 1910.08812 [cs]. URL: http://arxiv.org/abs/1910.08812 (visited on 03/08/2023). preprint.

[16] Peter Kán and Hannes Kafumann. 'DeepLight: Light Source Estimation for Augmented Reality Using Deep Learning'. In: *The Visual Computer* 35.6-8 (June 2019), pp. 873–883. ISSN: 0178-2789, 1432-2315. DOI: 10.1007/s00371-019-01666-x. URL: http://link.springer.com/10.1007/s00371-019-01666-x (visited on 03/08/2023).

[17] Adam Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] Hao Zhou, Xiang Yu and David Jacobs. 'GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE, Oct. 2019, pp. 7819–7828. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00791. URL: https://ieeexplore.ieee.org/document/9010991/ (visited on 03/06/2023).

[19] Robert Gützkow. *Render Normals in Camera Space*. Blender Stack Exchange. 10th Jan. 2020. URL: https://blender.stackexchange.com/a/157279/172211 (visited on 26/09/2023).

[20] Ben Mildenhall et al. 'NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis'. Version 2. In: (2020). DOI: 10.48550/ARXIV.2003.08934. URL: https://arxiv.org/abs/2003.08934 (visited on 01/06/2023).

[21] Mike Roberts et al. 'Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding'. Version 5. In: (2020). DOI: 10.48550/ARXIV.2011.02523. URL: https://arxiv.org/abs/2011.02523 (visited on 01/06/2023).

[22] Pratul P. Srinivasan et al. *NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis*. 7th Dec. 2020. arXiv: 2012.03927 [cs]. URL: http://arxiv.org/abs/2012.03927 (visited on 19/06/2023). preprint.

[23] Daniel Thul et al. 'Precomputed Radiance Transfer for Reflectance and Lighting Estimation'. In: *2020 International Conference on 3D Vision (3DV)*. 2020 International Conference on 3D Vision (3DV). Fukuoka, Japan: IEEE, Nov. 2020, pp. 1147–1156. ISBN: 978-1-72818-128-8. DOI: 10.1109/3DV50981.2020.00125. URL: https://ieeexplore.ieee.org/document/9320362/ (visited on 01/06/2023).

[24] Fangneng Zhan et al. *EMLight: Lighting Estimation via Spherical Distribution Approximation*. 20th Dec. 2020. arXiv: 2012.11116 [cs]. URL: http://arxiv.org/abs/2012.11116 (visited on 03/08/2023). preprint.

[25] Jonathan T. Barron et al. *Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields*. 13th Aug. 2021. arXiv: 2103.13415 [cs]. URL: http://arxiv.org/abs/2103.13415 (visited on 14/06/2023). preprint.

[26] Elena Garces et al. *A Survey on Intrinsic Images: Delving Deep Into Lambert and Beyond*. 7th Dec. 2021. arXiv: 2112.03842 [cs]. URL: http://arxiv.org/abs/2112.03842 (visited on 02/11/2023). preprint.

[27] Dor Verbin et al. 'Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields'. Version 1. In: (2021). DOI: 10.48550/ARXIV.2112.03907. URL: https://arxiv.org/abs/2112.03907 (visited on 01/06/2023).

[28] Alex Yu et al. *Plenoxels: Radiance Fields without Neural Networks*. 9th Dec. 2021. arXiv: 2112.05131 [cs]. URL: http://arxiv.org/abs/2112.05131 (visited on 03/06/2023). preprint.

[29] Xiuming Zhang et al. 'NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination'. In: *ACM Transactions on Graphics* 40.6 (Dec. 2021), pp. 1–18. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3478513.3480496. URL: https://dl.acm.org/doi/10.1145/3478513.3480496 (visited on 01/06/2023).

[30] Benjamin Attal et al. *Learning Neural Light Fields with Ray-Space Embedding Networks*. 10th May 2022. arXiv: 2112.01523 [cs]. URL: http://arxiv.org/abs/2112.01523 (visited on 07/08/2023). preprint.

[31] Partha Das, Sezer Karaoglu and Theo Gevers. 'Intrinsic Image Decomposition Using Physics-Based Cues and CNNs'. In: *Computer Vision and Image Understanding* 223 (Oct. 2022), p. 103538. ISSN: 10773142. DOI: 10.1016/j.cviu.2022.103538. URL: https://linkinghub.elsevier.com/retrieve/pii/S1077314222001163 (visited on 24/11/2023).

[32] Junxuan Li and Hongdong Li. *Self-Calibrating Photometric Stereo by Neural Inverse Rendering*. 15th July 2022. arXiv: 2207.07815 [cs]. URL: http://arxiv.org/abs/2207.07815 (visited on 03/08/2023). preprint.

[33] Linjie Lyu et al. 'Neural Radiance Transfer Fields for Relightable Novel-view Synthesis with Global Illumination'. Version 1. In: (2022). DOI: 10.48550/ARXIV.2207.13607. URL: https://arxiv.org/abs/2207.13607 (visited on 08/05/2023).

[34] S. Sharan Ranjit and Raj K. Jaiswal. 'Intrinsic Decomposition with Deep Supervision from a Single Image'. In: *Journal of King Saud University - Computer and Information Sciences* 34 (10, Part A 2022), pp. 8647–8657. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.09.006. URL: https://www.sciencedirect.com/science/article/pii/S1319157821002573.

[35] Wenqi Yang et al. *PS-NeRF: Neural Inverse Rendering for Multi-view Photometric Stereo*. 22nd Dec. 2022. arXiv: 2207.11406 [cs]. URL: http://arxiv.org/abs/2207.11406 (visited on 03/08/2023). preprint.

[36] Weicai Ye et al. 'IntrinsicNeRF: Learning Intrinsic Neural Radiance Fields for Editable Novel View Synthesis'. Version 2. In: (2022). DOI: 10.48550/ARXIV.2210.00647. URL: https://arxiv.org/abs/2210.00647 (visited on 01/06/2023).

[37] Changwoon Choi, Juhyeon Kim and Young Min Kim. *IBL-NeRF: Image-Based Lighting Formulation of Neural Radiance Fields*. 11th Sept. 2023. arXiv: 2210.08202 [cs]. URL: http://arxiv.org/abs/2210.08202 (visited on 02/11/2023). preprint.

[38] Matthew Tancik et al. ‘Nerfstudio: A Modular Framework for Neural Radiance Field Development’. In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH ’23. 2023.

[39] Marco Toschi et al. ‘ReLight My NeRF: A Dataset for Novel View Synthesis and Relighting of Real World Objects’. Version 1. In: (2023). DOI: 10.48550/ARXIV.2304.10448. URL: https://arxiv.org/abs/2304.10448 (visited on 01/06/2023).

[40] Zian Wang et al. ‘Neural Fields Meet Explicit Geometric Representation for Inverse Rendering of Urban Scenes’. Version 1. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023). DOI: 10.48550/ARXIV.2304.03266. URL: https://arxiv.org/abs/2304.03266 (visited on 01/06/2023).

[41] Frederik Warburg et al. ‘Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs’. Version 2. In: (2023). DOI: 10.48550/ARXIV.2304.10532. URL: https://arxiv.org/abs/2304.10532 (visited on 01/06/2023).