UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Utilisation de l'Apparence pour le Rendu et l'édition efficaces de scènes capturées

Siddhant Prakash

Inria Sophia Antipolis-Méditerannée

Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur
Dirigée par : George Drettakis
Soutenue le : 20 December 2022

Devant le jury composé de :
Tamy Boubekeur, Directeur, Adobe Research Paris
George Drettakis, Directeur de Recherche, Université Côte d'Azur, Inria Sophia Antipolis - Méditerranée
Nicolas Holzschuch, Directeur de Recherche, Inria Rhône-Alpes
Romain Pacanowski, Chargé de Recherche, Inria Bordeaux Sud-Ouest
Marc Stamminger, Professeur, Friedrich-Alexander-Universität Erlangen-Nürnberg
James Tompkin, Assistant Professeur, Brown University

Inria

Utilisation de l'Apparence pour le Rendu et l'édition efficaces de scènes capturées

---

Using Appearance to Efficiently Render and Edit Captured Scenes

Jury:

Président du jury / President of the jury
Nicolas Holzschuch, Directeur de Recherche, Inria Rhône-Alpes

Rapporteurs / Reviewers
Marc Stamminger, Professeur, Friedrich-Alexander-Universität Erlangen-Nürnberg
Nicolas Holzschuch, Directeur de Recherche, Inria Rhône-Alpes

Examinateurs / Examiners
Romain Pacanowski, Chargé de Recherche, Inria Bordeaux Sud-Ouest
James Tompkin, Assistant Professeur, Brown University
Tamy Boubekeur, Directeur, Adobe Research Paris

Directeur de thèse / Thesis supervisor
George Drettakis, Directeur de Recherche, Université Côte d'Azur, Inria Sophia Antipolis - Méditerranée

Acknowledgements

To begin with I'd like to express my gratitude towards my advisor, **George Drettakis**, without whom this thesis wouldn't be possible. George showed trust in me and offered me this PhD at a difficult time post my masters studies. His positive attitude, enthusiasm for innovation and a penchant for getting things done were my cornerstone throughout this PhD. For all his contributions in shaping my research career, I can't thank him enough.

I'm thankful for my awesome co-authors - **Gilles Rainer**, **Thomas Leimkühler**, and **Simon Rodriguez** for helping me through the projects when I got stuck. The resources and ideas that they brought to the table on a daily basis are a major factor in the success of this thesis. I'd like to especially thank **Adrien Bousseau** who has been a constant support both as a co-author and a guide whom I could turn to for advice on various aspects of the PhD life.

I thank the European Research Council for funding this PhD through the Advanced grant FUNGRAPH. I'm grateful to INRIA Sophia Antipolis-Méditerranée "Nef" computation cluster, the OPAL infrastructure from Université Côte d'Azur and EDSTIC for providing resources and support. I also extend my thank to the editors and anonymous reviewers for their insightful comments which helped improve the publications. I thank all my jury members who dedicated their invaluable time to my dissertation.

This acknowledgement will be incomplete without mentioning the GRAPHDECO team and the incredibly talented pool of people I had the good fortune to meet here. Thanks to **Felix Hähnlein**, **Stavros Diolatzis**, **Emilie Yu**, **David Jourdan**, **Georgios Kopanas**, and **Rada Deeb** for being my family away from home. I will always cherish the experiences and conversations we had over numerous hiking trips, sports, food & drinks, house parties and coffee breaks.

Finally, I want to thank my parents for their unwavering support to all my endeavours. They encouraged me to follow my interest in computer science and ensured I get whatever was required at every stage of my life. I'd also like to thank my sister **Ankita Prakash** for being the perfect role-model and leading the way for me to become a better person.

Résumé

L'informatique graphique a pour but de rendre des images de synthèse semblables à des photographies. Plusieurs algorithmes de rendu ont été développés au cours du dernier demi-siècle, principalement pour restituer des scènes à base d'éléments 3D créés par des artistes. Alors que les scènes initiales étaient assez simples, des représentations plus complexes de la géométrie, des matériaux et de l'éclairage ont été développés. Créer des scènes aussi complexes nécessite beaucoup de travail et de compétences de la part d'artistes 3D professionnels. Au même temps, les algorithmes de rendu impliquent des techniques de simulation complexes coûteuses en temps, pour résoudre le transport global de la lumière dans une scène.

Avec la popularité grandissante de la photo numérique, le rendu basé image (IBR) a émergé comme une alternative au rendu traditionnel. Avec cette approche, l'utilisation de photos comme données d'entrée est devenue beaucoup plus rapide que la génération de scènes classiques. Les algorithmes IBR se sont d'abord concentrés sur la restitution de scènes pour en permettre une exploration libre. Au fil du temps, les modèles de scène sont devenus plus complexes et l'utilisation d'un proxy géométrique inféré à partir d'images est devenue la norme. Aujourd'hui, l'utilisation d'un maillage reconstruit à l'aide des techniques Structure-from-Motion (SfM) et Multi-view Stereo (MVS) est courante en IBR, bien que cette utilisation introduit des artefacts importants.

Nous proposons d'abord **un nouvel algorithme de rendu basé image, *Hybrid-IBR*, qui se concentre sur le rendu de qualité et en temps interactif d'une scène capturée**. Nous étudions différentes faiblesses des travaux précédents et proposons un algorithme qui s'appuie sur ces travaux pour obtenir de meilleurs résultats. Notre algorithme se base sur l'apparence de la surface pour traiter les régions dont l'apparence dépend de l'angle de vue différemment des régions diffuses. Hybrid-IBR obtient des résultats favorables par rapport aux approches concurrentes pour une grande variété de scènes en termes de qualité et/ou de vitesse.

Bien que l'IBR soit une bonne solution de rendu, l'édition de celle-ci est difficile sans une décomposition en différents éléments : la géométrie, l'apparence des matériaux et

l'éclairage de la scène. Pour notre deuxième contribution, **nous estimons explicitement les paramètres de matériaux à *l'échelle de la scène* à partir d'un ensemble de photographies, pour permettre l'édition de la scène**. Alors que les solutions de photogrammétrie commerciales calculent la texture diffuse pour assister la création manuelle de matériaux, nous visons à créer *automatiquement* des atlas de texture de matériaux à partir d'un ensemble d'images d'une scène. Nous nous appuyons sur les informations fournis par ces images et les transmettons à un réseau neuronal convolutif pour obtenir des cartes de matériaux pour chaque vue. En utilisant toutes ces prédictions, nous créons des atlas de texture de matériau cohérents pour toutes les vues en agrégeant les informations dans l'espace texture. Nous démontrons l'utilisation de notre atlas de texture de matériaux généré automatiquement pour rendre des scènes réelles avec un changement d'illumination et avec des objets virtuels insérés.

L'apprentissage profond nécessite de grandes quantités de données variées. L'utilisation de données synthétiques est courante, mais l'utilisation du rendu traditionnel pour créer ces données prend du temps et offre une variabilité limitée. Nous proposons **une nouvelle approche basée sur le rendu neuronal qui apprend une représentation de scène neuronale avec paramètres variables, et l'utilise pour générer au vol de grandes quantités de données à un rythme beaucoup plus rapide**. Nous démontrons l'avantage d'utiliser le rendu neuronal par rapport au rendu traditionnel en termes de budget de temps, ainsi que pour l'apprentissage de tâches auxiliaires avec le même budget de calcul.

Mots-clés: rendu basé image - capture de matériaux - apprentissage profond - rendu neuronal

---

Abstract

Computer graphics strives to render synthetic images identical to real photographs. Multiple rendering algorithms have been developed for the better part of the last half-century. Traditional algorithms use 3D assets manually generated by artists to render a scene. While the initial scenes were quite simple, the field has developed complex representations of geometry, material and lighting: the three basic components of a 3D scene. Generating such complex assets is hard and requires significant time and skills by professional 3D artists. In addition to asset generation, the rendering algorithms themselves involve complex simulation techniques to solve for global light transport in a scene which costs more time.

As the ease of capturing photographs improved, Image-based Rendering (IBR) emerged as an alternative to traditional rendering. Using captured images as input became much faster than generating traditional scene assets. Initial IBR algorithms focused on creating a scene model using the input images to interpolate or warp them and enable free-viewpoint navigation of captured scenes. With time the scene models became more complex and using a geometric proxy computed from the input images became an integral part of IBR. Today using a mesh reconstructed using Structure-from-Motion (SfM) and Multi-view Stereo (MVS) techniques is widely used in IBR even though they introduce significant artifacts due to noisy reconstruction.

In this thesis we first propose **a novel image-based rendering algorithm, *Hybrid-IBR*, which focuses on rendering a captured scene with good quality at interactive frame rates**. We study different artifacts from previous IBR algorithms and propose an algorithm which builds upon previous work to remove such artifacts. The algorithm utilizes surface appearance in order to treat view-dependent regions differently than diffuse regions. Our Hybrid-IBR algorithm performs favorably against classical and modern IBR approaches for a wide variety of scenes in terms of quality and/or speed.

While IBR provides solutions to render a scene, editing them is hard. Editing scenes require estimating a scene's geometry, material appearance and illumination. As our second contribution **we explicitly estimate *scene-scale* material parameters from a set of**

captured photographs to enable scene editing. While commercial photogrammetry solutions recover diffuse texture to aid 3D artists in generating material assets manually, we aim to *automatically* create material texture atlases from captured images of a scene. We take advantage of the visual cues provided by the multi-view observations. Feeding it to a Convolutional Neural Network (CNN) we obtain material maps for each view. Using the predicted maps we create multi-view consistent material texture atlases by aggregating the information in texture space. Using our automatically generated material texture atlases we demonstrate relighting and object insertion in real scenes.

Learning-based tasks require large amounts of data with variety to learn the task efficiently. Using synthetic datasets to train is the norm but using traditional rendering to render large datasets is time consuming providing limited variability. We propose **a new neural rendering-based approach that learns a neural scene representation with variability and use it to generate large amounts of data at a significantly faster rate on the fly**. We demonstrate the advantage of using neural rendering as compared to traditional rendering in terms of speed of generating dataset as well as learning auxiliary tasks given the same computational budget.

Keywords: image-based rendering - material capture - deep learning - neural rendering

Contents

Chapter 1

Introduction

The goal of computer graphics (CG) is to generate synthetic images which resemble real photographs. Rendering is the process of generating the final image based on a scene description. A scene is composed of a geometric model, a material model and an illumination model. In order to achieve this goal, researchers have developed various rendering algorithms and techniques over the past $50+$ years. Rendering was formalized in a mathematical equation, known as the rendering equation, by Kajiya [1986]. In this thesis we propose methods of rendering and editing a scene efficiently by exploiting it's underlying material properties.

1.1 Context

Initially, the first systems were designed to create digital computer-generated imagery in 2D (Sutherland [1964]). Subsequently interest in 3D computer graphics propelled the field and the first ray-tracing algorithms were proposed by Appel [1968]. The ray

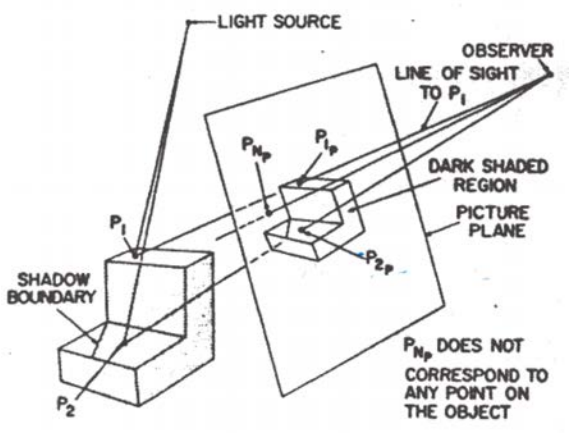


Figure 1.1: The ray tracing algorithm was first introduced by Appel [1968] which defined many key elements of a scene for rendering such as surfaces, light source, observer and picture plane or camera sensor.

Figure 1.2: Traditional rendering uses artist-generated scene assets such as a triangular mesh (left) for geometry, and complex material and illumination models to render the scene using a path tracer (right).

tracing algorithm defined the notion of a scene, which was composed of multiple *surfaces*, and rays from a *light source* were traced into the scene to render surfaces that were visible while culling those which were hidden from the *observer* on a *picture plane* or *camera sensor* (see Fig. 1.1). Initial scenes used for research were made up of simple geometric primitives such as planes and spheres. But as more advanced and complex algorithms were proposed the scene geometry, material and illumination models got increasingly complex such that dedicated 3D artist were required to fulfil the needs of a commercial graphics studio. Today a computer-generated movie studio such as Pixar Inc. have hundreds of CG artists as employees who work for hours to generate each frame of a 2-hour long movie at 30 frames per second with each frame taking upto $10 - 12$ hours of rendering[1]. We term this process of rendering a scene using artist generated 3D scene assets as traditional rendering.

Traditional rendering has two problems. The time taken to generate 3D scene assets is very high which makes the process of rendering a scene time-consuming (see Fig. 1.2). Additionally, traditional rendering typically involves complex simulation techniques such as Monte-Carlo rendering (Shirley [1991]; Arvo [1995]; Veach and Guibas [1995b]; Lafortune [1996]) and Metropolis light transport (Veach and Guibas [1997]). These simulations may take hours to render an image of a scene with sufficient quality.

Recently there has been a revolution in digital photography and capturing pictures of real environments has become increasingly fast and easy. This ease of acquiring photographs gave birth to another branch of rendering known as Image-Based Rendering (IBR) (Chen

[1]Why It Takes Pixar 3 Years To Render A Movie. `https://youtu.be/GBLPXM6mqBw`

[1995]; Levoy and Hanrahan [1996]; Debevec et al. [1996, 1998]; Shum et al. [2008]). The main idea behind IBR is to capture one or more photographs of a scene and use these *input viewpoints* to render the scene from novel viewpoints ultimately allowing free-viewpoint navigation in 3D. Typically IBR algorithms use cheap image processing operations such as warping and blending the input views into the novel views to achieve rendering. IBR's advantage is the simplification of content creation it achieves due to ease of capturing photographs as well as the speed-up in rendering due to the fact that complex light transport simulation is not required. IBR algorithms such as Soft3D (Penner and Zhang [2017]), Deep Blending (Hedman et al. [2018]), and using multi-plane images (Zhou et al. [2018]; Srinivasan et al. [2019]) achieves state-of-the-art in rendering casually-captured real scenes at a much faster rate than traditional rendering.

The idea of using images to factor a captured scene into its basic components of geometry, lighting and materials is known as inverse rendering (Marschner [1998]; Ramamoorthi and Hanrahan [2001]). Inverse rendering from captured images has gained huge popularity and importance over the past 20 years of computer graphics research. If we are able to factor a photograph into its components, we can utilize the components to re-render the scene with modified contents thus enabling editing of real scenes. Scene editing from a set of images is a very challenging task with multiple solutions proposed in the past two decades (Kholgade et al. [2014]; Zhang et al. [2016]).

In the last few years, a new branch of rendering has been invented by introducing neural networks to achieve high quality rendering and editing of scenes. Neural rendering (Tewari et al. [2020, 2021]; Wang et al. [2021]) is a very exciting and upcoming domain which was not present at the start of this thesis. The basic idea of neural rendering is to use a set of images to learn the scene (and its underlying properties) with a neural network such that one can re-render the scene as desired by evaluating the learned network. We have developed solutions which utilize all three forms of forward rendering to achieve our goals of rendering and editing real scenes efficiently.

1.2 High-level Goals and Research Questions

Traditional rendering achieves high quality of photorealism but at the cost of time and computational resources. Creating the 3D scene assets required for high quality rendering takes a large amount of skill and familiarity with the scene modelling process.

The rendering algorithms themselves involve solving complex simulations which requires huge computational power often provided by specialized hardware such as Graphics Processing Units (GPUs). One of the goals of this thesis is *to reduce the time required in generating assets as well as rendering scenes.*

Image-based Rendering (IBR) solves the problem regarding the time taken to generate scene assets. Due to the ease of taking photographs we can easily capture an entire scene with the help of commercial hand-held cameras in a few minutes. With the recent advancement in automatic reconstruction techniques, we can generate a 3D mesh of the scene within a few minutes or hours depending on number of images captured. The mesh along with the images are used to render scenes efficiently. The meshes reconstructed automatically are not perfect and reduce the quality of rendering due to defects such as holes, irregular surfaces and floating geometry. Researchers have tried to get rid of such geometric artifacts over the years and we follow the same approach in the solutions we propose to improve upon rendering quality. While IBR algorithms are fast, the state-of-the-art algorithms are still not fast enough to enable real-time rendering. We also aim *to improve the rendering speed of IBR algorithms to enable real-time rendering even on hardware-constrained devices such as a laptop.*

IBR algorithms have used scene geometry as an input to improve rendering quality but surface appearance has still not been exploited properly. Works such as Sinha et al. [2012]; Kopf et al. [2013]; Rodriguez et al. [2020b] merely touch upon this aspect without explicitly recovering the underlying surface property. At the same time, these works prove that knowledge of surface appearance can help us design better IBR algorithms. Hence another of our goals is *to try and use surface appearance to our aid in exploration of an ideal IBR algorithm.*

One of the disadvantages of IBR algorithms is that the scenes are not editable. Once the input images are captured, we cannot render the scene with edited components such as novel lighting or different materials. To do so, we need to factor the scene from the images into its basic components of geometry, material and illumination. While we have solutions for extracting geometry of a scene using Structure-from-Motion (SfM) and Multi-view Stereo (MVS) techniques (Schönberger and Frahm [2016]; Schönberger et al. [2016]), we have limited solutions for scene-scale material recovery often requiring user intervention. In this thesis we also explore extracting surface material properties

from images automatically on a scene-scale using the geometry recovered from previous solutions. We exploit learning-based solutions which has been recently shown to be very useful in material capture tasks on small-scale surfaces or patches (Deschaintre et al. [2018, 2019]). Our main aim is *to automatically create a model of the scene from photographs which can be edited and re-rendered with a different scene configuration using a traditional path tracer.*

Learning-based approaches have taken the field of computer graphics to new levels in recent years. A number of problems are being solved by learning-based methods achieving state-of-the-art results. One of the basic caveat of most learning-based approaches is their need for data. They require large amounts and variety in data to achieve generalized solutions valid for a large set of use cases. Unfortunately, acquiring ground truth real data is virtually impossible for many tasks such as material capture. To this end, synthetic data can help fill those gaps to obtain large amount of paired realistic data.

Obtaining large quantities of data as required by learning-based approaches is not trivial. Generating data using traditional rendering is time-consuming as we discussed previously. Ideally we require highly photorealistic synthetic data to reduce the domain gap between real and synthetic scenes on which our learning task will be tested. To reduce complexity and speed up the process of obtaining synthetic data we *explore neural scene representations to obtain large amounts of data with sufficient variety in a fast and efficient manner.*

We believe that obtaining synthetic data using neural rendering can generate more data with variety than traditional rendering within the same computational budget. We also study data generation on-the-fly interleaved with training a network on a given task. It is interesting to explore if using neural scene representations to generate data for learning tasks can benefit the task at hand. Generating more data for training within a given computational budget should already help the learning task in faster generalization. Additionally, if the training samples generated online provide samples which aid in learning the task at hand more efficiently, there may be no need for curating a dataset for particular tasks beforehand. We explore if such an *active exploration* of data space is possible during training and if so how can neural rendering help with it.

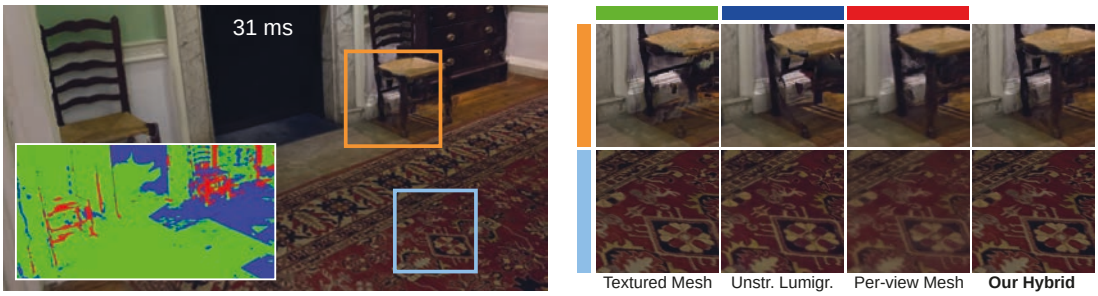


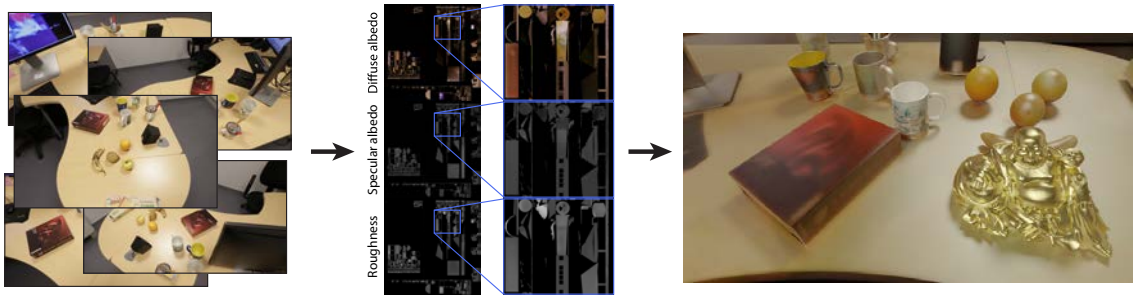
Figure 1.3: Our method analyzes the geometric and photometric consistency of a casually captured scene and for each novel-view pixel selects the best rendering algorithm (color mask, Red: Per-view Mesh; Blue: Unstructured Lumigraph; Green: Textured Mesh). We address color seams, view-dependent effects, missing geometry (orange inset, top), and texture sharpness (light blue inset, bottom), while retaining interactive frame rates.

1.3 Overview

In this thesis we present three contributions. First, we develop a novel IBR algorithm called Hybrid Image-Based Rendering (Hybrid-IBR). Many solutions exist for free-viewpoint rendering which are either fast or focus on rendering quality. Solutions which are fast, such as a textured mesh or the Unstructured Lumigraph Rendering (ULR) (Buehler et al. [2001]) often suffer with rendering quality due to geometric artifacts such as ghosting and aliasing or omit view-dependent effects. Other classes of algorithms which focus on rendering quality, such as the per-view mesh based solutions of Hedman et al. [2016, 2018], are costly to render and can not run interactively on resource constrained devices such as a laptop. Our solution combines the strengths of previous IBR algorithms and strikes a balance between the rendering quality and speed to enable good quality rendering at interactive frame rates. We discuss the algorithm in the Chapter 3 of this thesis. Fig. 1.3 shows the advantage of rendering a novel view using our approach compared to previous methods.

For our Hybrid-IBR algorithm we obtain a rough estimate of the underlying surface appearance by computing *photometric uncertainty* between multiple views. This gives us an idea about the material of the underlying surface but does not provide us with explicit material parameters.

To this end as our second contribution, we attempt to solve the hard probelm of recovering underlying *scene-scale* material properties explicitly given multiple views of the scene in the next chapter. We use a learning-based approach to predict the material parameters for



(a) Multi-view capture (b) SVBRDF texture atlas (c) Physically-based rendering

Figure 1.4: We present a method which takes multiple photographs of a scene as input (a) and predicts surface materials in the form of material maps corresponding to each input view. Merging these image-space predictions in texture space yields a texture atlas (b) that can be mapped onto retopologized geometry to produce *digital 3D assets* ready for full physically-based rendering, i.e., rendering from any viewpoint, changing/adding lights and objects, for example in (c) the golden statuette and the white mug have been added to the scene that has modified lighting and is rendered from a viewpoint not in the input.

each point of the scene by exploiting multi-view observations of the given point. To train our network we propose a new scene-scale dataset by rendering artist-generated scenes under different configurations of geometry, materials and lighting. We demonstrate the effectiveness of our recovered material texture atlases by re-rendering real scenes with modified contents. Our proposed solution is demonstrated in Fig 1.4.

The dataset we used for training our neural network for material estimation was rendered using a traditional rendering pipeline. Although of high quality, the amount of paired images we could generate within a given time budget was limited. To improve this aspect, in our third contribution we investigate using neural scene representations to generate datasets with variation in large quantities and a short timeframe. Our proposed solution is able to render a wide variety of images with full global illumination at a speed faster than traditional rendering methods. We provide initial experiments and results to show that using neural approaches to rendering datasets helps improve learning tasks over traditionally generated datasets within the same total computational budget.

1.4 Contributions

The thesis was funded by the ERC Advanced grant FUNGRAPH (grant number 788065) which led to two journal publications and a third ongoing project. Our novel Hybrid

IBR algorithm (Prakash et al. [2021]) was published in the Proceedings of the ACM in Computer Graphics and Interactive Techniques in April 2021 and the scene-scale material capture method (Prakash et al. [2022]) was published in the journal Computers & Graphics in October 2022.

Chapter 2

Previous Work

We look at some of the most relevant previous work in this chapter. First we discuss rendering algorithms categorized by the key component used as traditional rendering i.e., using traditional 3D scene assets, image-based rendering i.e., using captured images only or neural rendering i.e., using neural networks. Then we take a brief look at inverse rendering from captured images in the context of scene editing. We follow that up by taking a look at material appearance, its definition, representation and capture of real world materials. Finally we shed light on some recent neural scene representation methods and its use in rendering and editing of real scenes.

2.1 Rendering

In this section we give an overview of the advances in rendering algorithms over the past $50+$ years. First we briefly look at the history of physically based rendering noting major breakthroughs in traditional rendering algorithms. Then we discuss the development of image based rendering (IBR) algorithms followed by the very recent breakthroughs in neural rendering.

2.1.1 Traditional rendering

During the 70's and 80's first algorithms for rendering were proposed. At that time most researchers tried solving the fundamental problems of visibility and scene representations. In his early paper Appel [1968] proposed the first ray tracing algorithm which defined key scene elements such as cameras, light source, ray-surface intersection etc. Further advancement using ray tracing algorithms saw rendering of transparent surfaces using an algorithm proposed by Kay and Greenberg [1979]. The paper that revolutionized the field of ray tracing was by Whitted [1980] which proposed the first recursive ray-tracing algorithm and showed never before seen phenomena in a rendered image with global illumination effects simulating lighting distribution in a scene (see Fig. 2.1).

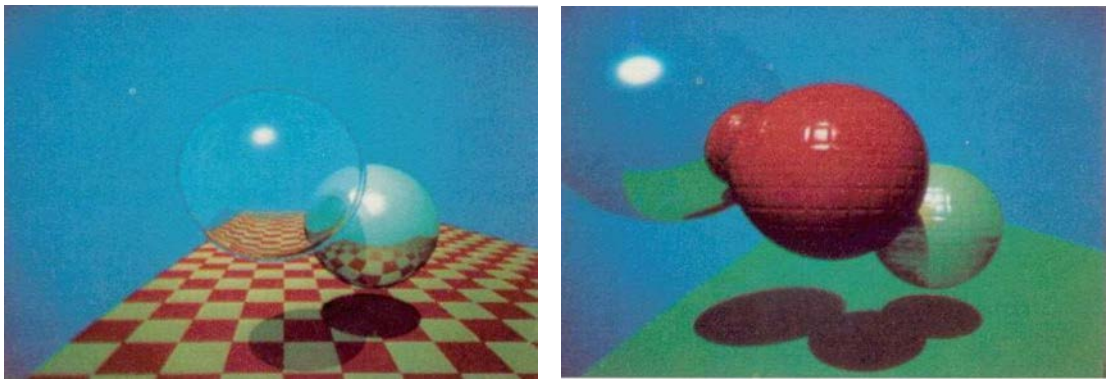


Figure 2.1: Images generated with recursive ray tracing algorithm of Whitted [1980].

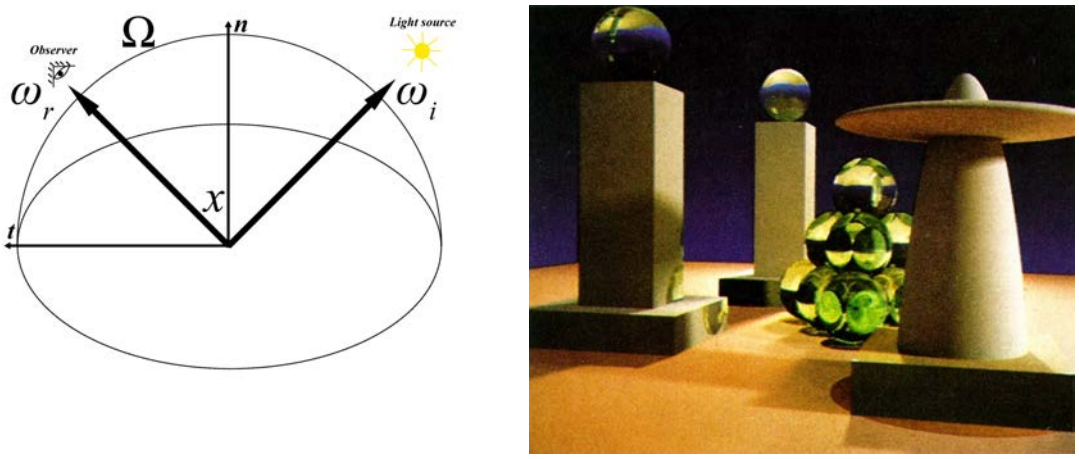


Figure 2.2: Kajiya [1986] proposed the rendering equation which describes the outgoing radiance at a surface point **x** (left) and solved it using Monte-Carlo integration to render an image (right).

Interest in physically based rendering led to the use of Monte-Carlo integration techniques with ray tracing to simulate global illumination accurately. In his seminal paper, Kajiya [1986] introduced path tracing which described the rendering equation and how to solve this equation using Monte-Carlo simulation to render an image with full global illumination. The equation computes the outgoing radiance L_o at a surface point $\mathbf{x}$ viewed from the direction ω_o and is given by

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i.\mathbf{n}) \partial\omega_i \tag{2.1}$$

Figure 2.3: Eric Veach proposed the Metropolis Light Transport (Veach and Guibas [1997]) and other Monte-Carlo simulation based algorithms to contribute significantly in advancement of state-of-the-art in practical physically-based rendering for VFX industry.

Based on the equation, a 3D scene can be defined as a collection of surface points $\mathbf{x}$ with normal $\mathbf{n}$ at each point exhibiting material reflectance f_r given by the Bi-directional Reflectance Distribution Function (BRDF). The outgoing radiance L_o is sum of the reflected and the emitted radiance L_e if the surface emits light. The reflected radiance is computed as the incoming radiance L_i incident at the surface from direction ω_i multiplied with the BRDF f_r and the cosine of the incident angle integrated over the unit hemisphere Ω centered around the normal at the surface which contains all possible values of ω_i. At the time of its introduction the equation was costly to solve, taking hours to compute for a single 256x256 image (see Fig. 2.2).

Subsequent work by Shirley [1991] and Arvo [1995] in their respective dissertations advanced the computation of light transport using Monte-Carlo techniques both in terms of speed and quality. A key contribution in the wide acceptance of Monte-Carlo simulation based techniques for practical rendering in VFX industry was provided by Veach [1998] in his seminal papers on bi-directional path tracing (Lafortune and Willems [1993]; Veach and Guibas [1995a]), Multiple Importance Sampling (Veach and Guibas [1995b]), and Metropolis Light Transport (Veach and Guibas [1997]) (see Fig. 2.3). These works serve as

the building blocks of many commercial and academic physically based rendering system such as RenderMan (Apodaca and Mantle [1990]; Apodaca et al. [2000]; Christensen et al. [2018]), Arnold (Georgiev et al. [2018]), V-Ray (Chaos Group [2017]), PBRT (Pharr et al. [2016]), Mitsuba (Jakob [2010]; Nimier-David et al. [2019]), and Falcor (Kallweit et al. [2022]).

Despite producing synthetic images which are highly photorealistic, traditional rendering algorithms are still slow and require highly complex scene components which takes hours if not days to create by professional artists. To render photorealistic images as can be seen in a 3D animated movie, it is highly important to model the scene's 3D geometry, lighting and particularly the appearance accurately. Increasingly complex appearance models such as the Disney BRDF (Burley and Studios [2012]) have been proposed to model materials such as glass, sand, clothes, hair, wood, bricks etc. With increase in complexity it becomes increasingly hard to model such geometry and appearance. Most 3D animation studios employ highly skilled 3D artists who spend hours generating such high quality assets for traditional rendering pipelines. In this thesis we want to alleviate this problem and propose solutions to capture high quality material assets which are compatible with conventional path tracers, using photographs or video.

While modelling the 3D assets takes time and skill, rendering algorithms themselves employ complex simulation techniques which are time consuming. In typical Monte-Carlo simulation techniques, we sample multiple rays at a time to compute the color per pixel. Since this process is compute intensive, we encourage parallelization and use hardware which can speed up this task. If we query with low samples per-pixel we introduce noise in the rendered image due to incomplete sampling of the scene. With increase in samples the noise reduces but also increases the time taken to render images. We try to address this issue by using alternative mode of rendering such as image-based or neural rendering methods.

2.1.2 Image-based Rendering (IBR)

Image-based rendering (IBR) was primarily developed to provide a cheap alternative to traditional rendering by using captured photographs to re-render the scene from novel views. Initially Chen and Williams [1993]; Chen [1995]; McMillan and Bishop [1995] introduced the idea of view interpolation between captured or rendered images to synthesize novel views. The first IBR algorithms proposed by Levoy and Hanrahan

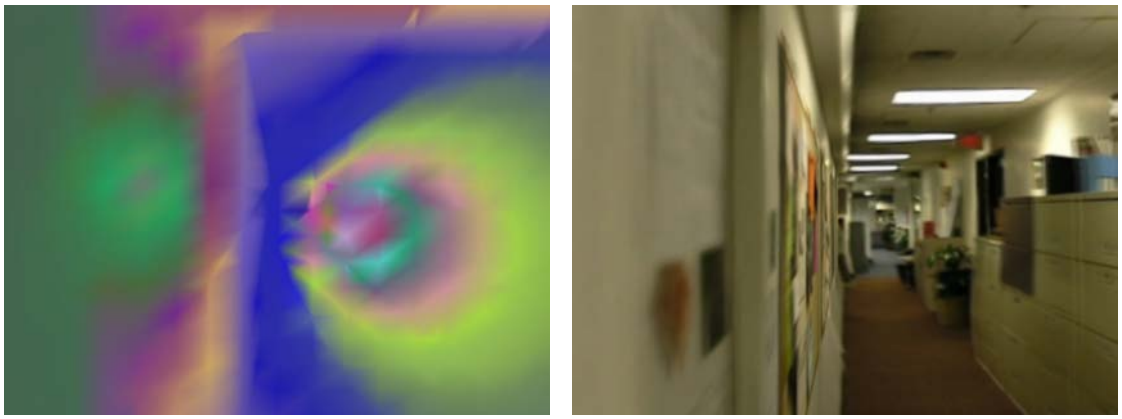


Figure 2.4: The Unstructured Lumigraph Rendering (ULR) proposed by Buehler et al. [2001] computed blend weights per input view (left) to warp and blend the input views into a novel view (right).

[1996]; Gortler et al. [1996] used captured photographs as a slice of the plenoptic function to create a model of the scene and used the model to render it from novel views. While images provided the base color for final blending, using geometry for interpolation was first introduced by Debevec et al. [1996, 1998] which projected the images on to a mesh built with user input and then blend images per texel to efficiently render view-dependent effects in novel views allowing sparse capture.

Subsequent advances in IBR improved either the quality of rendered novel views or the speed of rendering by using unstructured views acquired from casual capture (Buehler et al. [2001]) or via crowdsourcing on the internet (Snavely et al. [2006]). The Unstructured Lumigraph Rendering (ULR) proposed by Buehler et al. [2001] enabled IBR with casually captured images by using an automatically generated geometric proxy to compute the blend weights for blending the input images (see Fig. 2.4). Furthermore, the paper introduced a set of goals that IBR algorithms should follow for efficient image-based rendering. The goals included the use of geometric proxies, unstructured input, resolution consistency etc. as well as real-time runtime as one of the desired properties of IBR.

With the advances in Structure-from-Motion (SfM) and Multi-view Stereo (MVS) algorithms, use of explicitly generated geometry became more viable for IBR. Snavely et al. [2006, 2008] proposed a system for camera calibration and viewing of rendered images to enable virtual tours of captured locations through crowd-sourced images. The images obtained through crowd-sourcing from the internet were highly unstructured

Figure 2.5: The first interactive Image Based Rendering (IBR) system, PhotoTourism, was proposed by Snavely et al. [2006] which allowed a user to interact and navigate smoothly between photos of a landscape collected by crowdsourcing.

and the system provided a robust solution to calibrate such images (see Fig. 2.5). A line of subsequent works tried improving IBR by either rectifying geometric errors or using different geometric proxies.

Although the global meshes obtained from MVS were good, they still have geometric errors and a solution was proposed by Eisemann et al. [2008] to model floating geometry by using optical-flow based warp and soft-visibility. Goesele et al. [2010] explored the use of ambient point clouds to render the scene. While these algorithms improved the rendering quality, the cost of preprocessing the algorithms to obtain high quality geometric proxy increased.

Despite these improvements, the global mesh still suffers from poor reconstruction in many parts of the scene, e.g., non-diffuse and transparent surfaces that are not photo-consistent, or small, thin and/or repetitive structures that are hard for MVS. One class of solutions to these geometric issues is to use *per-view* information for each input photo. Superpixel-based depth map refinement and synthesis was used by Chaurasia et al. [2013] together with shape-preserving warps to the novel views, while per-view meshes from refined depth maps were used in Inside Out (Hedman et al. [2016]) to rectify reconstruction errors (see Fig. 2.6). In both cases, geometric details incorrectly reconstructed or even missing in the global mesh are corrected. And in terms of speed, both algorithms run in real-time on a system for IBR (Bonopera et al. [2020]) while the pre-processing can take a few minutes to hours based on the number of images.

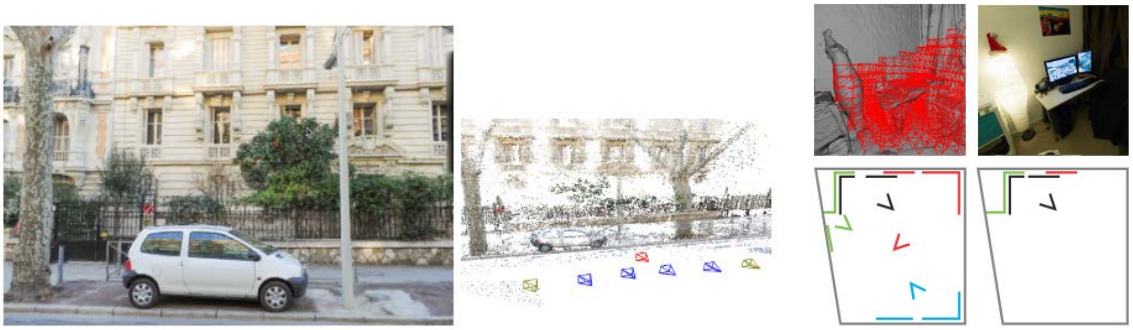


Figure 2.6: To overcome poor reconstruction quality of MVS meshes, *per-view* solutions were proposed by Chaurasia et al. [2013] (left, center) and Hedman et al. [2016] (right).

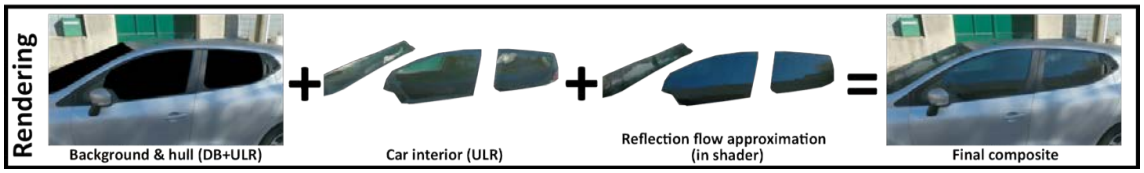


Figure 2.7: Revisiting the idea of layered rendering, Rodriguez et al. [2020b] proposed a solution to render the reflections on the mirrors of a car and blending it with background rendered with previous IBR algorithms.

Another class of algorithm attempted IBR based on the underlying surface property of the scene. Methods such as Sinha et al. [2012]; Kopf et al. [2013] used layered rendering to separate out reflection layers for reflective and glossy surfaces from the diffuse base layer of the surface enabling rendering of reflective objects. Similar to the idea of volumetric rendering, Penner and Zhang [2017] reconstructed a soft 3D volume of the scene using the input images, which was then queried at runtime to render novel views.

Recently the idea of layered rendering was revisited by Rodriguez et al. [2020b] in which semantic masks were obtained from the input images and used to accurately synthesize the reflective layers for the windows of cars. The reflective layers were rendered separately and then blended with the background which were rendered by previous IBR algorithms such as ULR (Buehler et al. [2001]) and Deep Blending (Hedman et al. [2018]) (see Fig. 2.7). Additionally, the paper also proposed a mesh-refinement technique to obtain a clean mesh of the cars free from bumps and holes found in the raw MVS mesh. Taking the idea of layered rendering forward, Xu et al. [2021] proposed a reflection decomposition algorithm which reconstructs the front and back layer of a reflective surface. They used a convolutional neural network (CNN) based super-resolution network to render

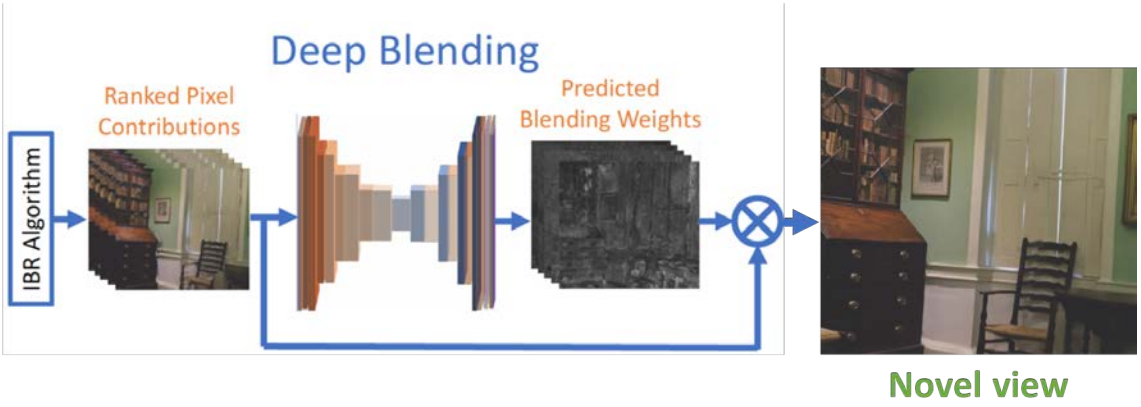


Figure 2.8: Deep Blending (Hedman et al. [2018]) was one of the earliest learning-based IBR approach which learned blend weights to warp and blend *input mosaics* into novel-view using per-view meshes.

high-resolution images from low-resolution textures with high-frequency reflections.

2.1.3 Neural Rendering

Towards the end of last decade *neural rendering* (Tewari et al. [2020]) emerged as a new rendering paradigm in computer graphics which involved solving problems with the help of neural networks. One of the earliest neural rendering algorithms was Deep Blending (Hedman et al. [2018]) which used neural networks to predict the blend weights with which the warped input images need to be blended to synthesize a novel view (see Fig. 2.8). The algorithm used per-view meshes reconstructed using depth maps which were refined with two MVS solutions and heuristics to generate per-pixel ranked mosaics which served as input to the network. The advantage of the algorithm was that it got rid of almost all geometric artifacts caused due to erroneous MVS geometry and learnt to fill the gaps with plausible colors based on the neighbouring pixels.

Using the idea of tile-based rendering proposed by Hedman et al. [2016, 2018], Wu et al. [2022] introduced two multi-layered perceptron (MLP) in each tile to render the diffuse color and view-dependent highlights using dedicated MLPs. The final color was obtained by blending the individual color predicted per-tile.

Thies et al. [2018] also attempts to separate diffuse and specular components of a scene and learns the view-dependent effects explicitly. The view dependent effects are predicted by a network and combined with diffuse textures in the final view using another network

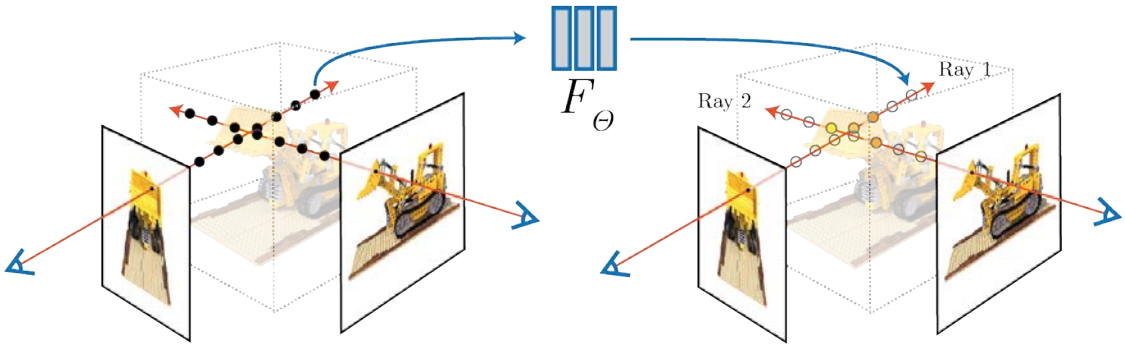


Figure 2.9: Learning an implicit scene representation called Neural Radiance Field (NeRF) from a set of input images, Mildenhall et al. [2020] achieved state-of-the-art in novel-view synthesis without using explicit geometry such as triangular meshes or point clouds.

which learns how to combine appearance of captured images instead of the underlying appearance of objects. Using the MVS mesh as a base, Riegler and Koltun [2020] used a recurrent neural network to learn warping and blending input views to novel views separately. Follow up work by Riegler and Koltun [2021] also used the MVS mesh on which directional feature vectors were learnt on each 3D point of the scene and new feature vectors were generated for rays from target views which was then decoded to render the novel view.

Using point clouds instead of the MVS mesh, Kopanas et al. [2021] proposed a new point-based differentiable renderer with per-view optimizations to achieve state-of-the-art in novel view synthesis in real time. In a concurrent work, using a multi-layered perceptron (MLP) and a ray transformer, Wang et al. [2021] learns a generalized view interpolation function over a continuous 5D domain (3D spatial location and 2D view directions) which is used to render novel-views from novel scenes using classical volume rendering.

For the task of novel view synthesis, a different school of thought emerged in which researchers learn implicit scene representations by feeding images of a scene to the neural network. These solutions are not general and learns only one scene at a time using a neural network. Some works that follow this line include learning local light fields (Mildenhall et al. [2019]), neural textures (Thies et al. [2019]), gradient descent (Flynn et al. [2019]), or volumetric representations such as multi-plane images (Zhou et al. [2018]; Srinivasan et al. [2019]), and 3D volumes or voxels (Meshry et al. [2019]; Lombardi et al. [2019]; Sitzmann et al. [2019]).

Most successful amongst this line of works is the work of Mildenhall and colleagues called Neural Radiance Fields (NeRF) (Mildenhall et al. [2020]; Martin-Brualla et al. [2021]) which uses a multi-layered perceptron (MLP) neural network with positional encodings to learn the radiance field of a scene composed of the color and density at each surface point (see Fig. 2.9). The radiance field is queried at render time using volumetric ray marching to integrate the points along a given ray and obtain the final output color. Numerous follow up works which try to improve the rendering quality (Sitzmann et al. [2021]) or the speed of rendering (Reiser et al. [2021]) and even training time of the network (Müller et al. [2022]) have been proposed. This work is advancing the state-of-the-art in novel view synthesis at a rapid pace.

While traditional rendering has come a long way following the initial breakthroughs of Shirley [1991]; Arvo [1995]; Veach [1998], it is still too costly to generate high-quality assets and render them even with state-of-the-art path tracers such as Falcor (Kallweit et al. [2022]). IBR provides a faster alternative to rendering realistic scenes but the artifacts due to geometry still persists even with solutions such as Chaurasia et al. [2013]; Hedman et al. [2016] targeting these issues. In this thesis we expand upon the idea of layered rendering (Kopf et al. [2013]; Rodriguez et al. [2020b]) for image-based rendering and propose a Hybrid-IBR algorithm which builds upon previous IBR solutions such as textured mesh (Debevec et al. [1998]), ULR (Buehler et al. [2001]) and Deep Blending (Hedman et al. [2018]).

Traditional rendering is used in this thesis to generate a synthetic dataset which is used to train a neural network for material estimation. We also use traditional rendering, i.e. a path tracer (Jakob [2010]), to re-render scenes with modified content using the material textures obtained by our method enabling editing of captured scenes. We further study neural rendering (Diolatzis et al. [2022]) for on-the-fly dataset generation in order to build a better dataset of synthetic images for learning tasks at a faster rate and with more variability than traditional rendering (Nimier-David et al. [2019]).

2.2 Inverse Rendering

In the previous section we reviewed works that used images directly for rendering a scene from novel views. While the rendering quality is good, the scenes cannot be edited at the time of rendering. To re-render the scene with edited components and for general scene

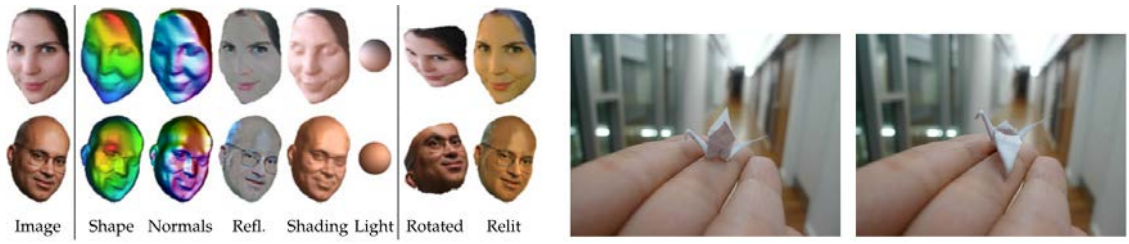


Figure 2.10: Inverse rendering is the process of recovering the scene's shape, material reflectance and illumination properties from a photograph as achieved by Barron and Malik [2015] (left). This is very useful for scene editing as the image can be re-rendered with edited content as shown by Kholgade et al. [2014] (right).

understanding, the past couple of decades has seen a number of researchers proposing algorithms for *inverse rendering* from captured images.

Inverse rendering, the process of recovering geometry, materials and/or illumination of a scene from images, is a long-standing problem in computer graphics and vision. One of the earliest scene-scale inverse rendering work was by Yu et al. [1999] in which a set of photographs was captured under known illumination condition and inverse radiosity was used to solve for illumination independent reflectance property of all surfaces which was provided as input to the method. In their classic paper Ramamoorthi and Hanrahan [2001] introduced a solution from signal processing perspective famously establishing the reflected light field observed by a sensor as the convolution of incoming light and BRDF at the surface. Applying their theory, they showed that complex lighting and BRDF can be factored out from a set of only 3 photographs given robust estimation of underlying surface. This led to many inverse rendering solutions trying to jointly estimate the underlying geometry with the material and lighting in form of reflectance and shading.

One of the most prominent works is the method by Barron and Malik [2015] which successfully factorize shape, reflectance and shading of a range of objects from a single image by formulating the problem as an optimization to recover the most likely set of parameters that explains the observed color intensity. Karsch et al. [2011, 2014] targeted virtual object insertion in a photograph by creating a 3D scene model from the photograph using human annotations or data-driven approaches to infer depth and diffuse reflectance in image space which is further optimized to jointly estimate illumination of the scene. Given a single photograph of a scene with a 3D object, Kholgade et al. [2014] enabled

realistic manipulation of the object in 3D space and produce novel views within the image with 6DoF using similar stock 3D model downloaded from the internet (see Fig. 2.10). Zhang et al. [2016] also showed applications of emptying and augmenting a room with 3D assets by inferring a diffuse albedo and direct illumination given a set of images and corresponding depth.

The reflectance model estimated by these methods are not very expressive which leads to unrealistic material editing and relighting of the scene especially from unobserved views. Many works perform intrinsic image decomposition to recover the low frequency diffuse albedo and the high frequency specular component from single image. Once obtained the decomposed layers can be used to relight the image realistically. While many solutions such as Li and Snavely [2018]; Meka et al. [2018] exists, the relighting quality still suffers due to inferior material estimation. Thus it becomes imperative to model appearance of the scene with high quality to re-render the scene with edited components.

Most inverse rendering methods attempt to recover 3D scene properties including geometry, material and illumination from a single image. In this thesis we propose to recover only material properties of the scene using multiple observations instead of a single image. We recover material parameter of a physically-based material model for the underlying surface to model the appearance of real world materials more realistically.

2.3 Material Representation & Capture

In this section we give an overview of surface material properties used for rendering. First we discuss material representation and then look into increasingly complex material models developed over the years. Then we look at some methods of traditional material acquisition followed by the most recent advancements in material capture with learning and optimization-based methods.

2.3.1 Material Representation

As we saw earlier, material reflectance is described by the Bi-directional Reflectance Distribution Function (BRDF) in the rendering equation (Eq. 2.1). The BRDF describes how much of a ray of light incident at a surface will be reflected by the surface in a given outgoing direction. The BRDF was proposed by Nicodemus [1965]; Nicodemus et al. [1977] as a specialized version of the Bi-directional Scattering-Surface Reflectance-

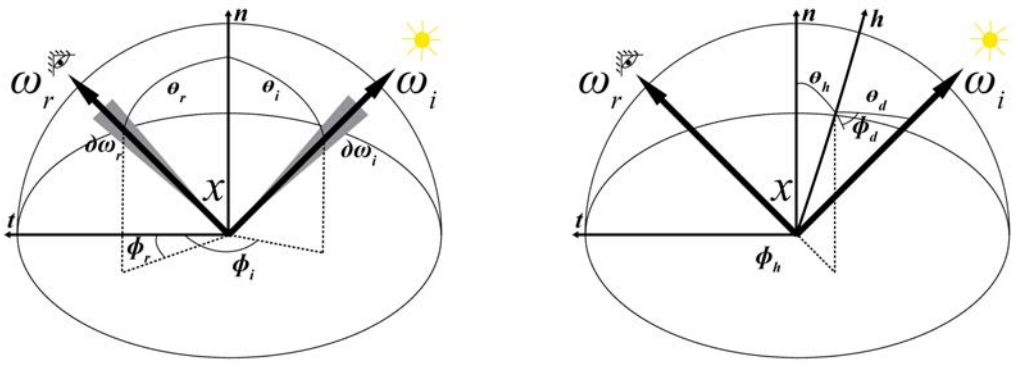


Figure 2.11: Geometry of surface for BRDF parameterization. Figure adapted from Guarnera et al. [2016].

Distribution Function (BSSRDF). The BRDF is described for opaque surfaces which do not exhibit sub-surface scattering. In this thesis we restrict our discussion to BRDFs representing opaque materials only. The BRDF assumes light entering a surface at a point also leaves the surface at the same point.

The BRDF is defined as

$$f_r(\omega_i, \omega_r) = \frac{\partial L_r(\omega_r)}{\partial E_i(\omega_i)} = \frac{\partial L_r(\omega_r)}{L_i(\omega_i) cos\theta_i \partial \omega_i} \tag{2.2}$$

where ω_i indicates the incident (i) direction and ω_r indicates the reflected (r) direction. E_i is incident irradiance (or incoming energy per unit surface area), L_i is the incident radiance (or incoming energy per unit area per unit solid angle) also known as field radiance and L_r is the reflected radiance (or outgoing energy per unit area per unit solid angle) also known as surface radiance or scene radiance (see Fig. 2.11). The units of BRDF are inverse steradians $[1/sr]$.

To represent BRDF over a surface, the Spatially Varying BRDF (SVBRDF) was proposed which adds extra parameters with respect to the BRDF to account for the location of the BRDF on the surface. SVBRDF models are used frequently to represent flat and opaque surfaces. Although SVBRDFs can model surfaces at a coarse scale, many fine-grained

reflectance effects such as inter-reflections, self-occlusion and self-masking are difficult to model using them. To model such mesoscopic effects Dana et al. [1999] introduced the Bi-directional Texture Function (BTF) which is an image based representation used to describe fine-scale appearance of rough surfaces.

BTFs require large amounts of data to sample the surface and thus have high storage demands. Over the years many researchers have tried to come up with different models to efficiently store and represent both BTF and BRDF data. It must be noted that no universal model which can represent every material exists yet. We now discuss some key BRDF models developed throughout the history of computer graphics.

2.3.2 BRDF Models

Early BRDF models tried to describe the appearance of simple isotropic surfaces. For isotropic surfaces the appearance does not depend on the orientation of the surface, i.e. the reflectance properties are invariant to rotations of the surface around the normal. Some examples of isotropic surfaces are plastic, glass, smooth metal etc. Anisotropic surfaces vary their reflectance based on the surface orientation, i.e., a change in orientation of the surface around the normal changes the reflectance behaviour. Examples of such surfaces are wood, brushed metal, copper, ice etc.

Many researchers have developed different ways to represent BRDFs. Early works try to fit *analytical models* to describe captured reflectance data (Phong [1975]; Blinn [1977]; Ward [1992]; Ashikhmin and Shirley [2000]). The representations are highly approximate and do not reproduce real world materials very well. *Physically-based models* have been proposed to model the Physics and Optics of rough surfaces at a fine scale (Cook and Torrance [1982]; Oren and Nayar [1994]; Walter et al. [2007]). These representations have a similar mathematical formula and are able to represent a range of materials with microsurface geometry accurately. In other works, measured BRDF data are tabulated and stored which can be further interpolated to model uncaptured BRDF. The challenge is to compress and store such large amount of data efficiently. This led to *data-driven models* used to approximate the BRDF data and interpolate or extrapolate the data to fill gaps in measurements (Matusik et al. [2003a,b]; Pacanowski et al. [2012]; Rainer et al. [2019, 2020]).

The most simple BRDF model is the Lambertian model which represents all diffuse

surface without any specular component. It is given by: $f_r(\omega_i, \omega_r) = \rho/\pi$ where ρ is a constant based on the material. One of the earliest models which was able to represent non-Lambertian materials was proposed by Phong [1975]. Due to the computational convenience of computing this model it was widely adopted at the time. It is given by: $f_r(\omega_i, \omega_r) = k_s(\omega_r.\mathbf{r}_{\omega_i})^n$ where k_s is a specular constant in the range of $[0, \inf]$, $\mathbf{r}_{\omega_i}$ is the mirror reflection vector of ω_i and n controls shape of the specular lobe. To make the computation efficient Blinn [1977] proposed a modification to this model by using the half-angle vector $\mathbf{h}$ and the normal at the surface $\mathbf{n}$ instead of the reflection vector $\mathbf{r}_{\omega_i}$. The Blinn-Phong model is given by: $f_r(\omega_i, \omega_r) = k_s(\mathbf{n}.\mathbf{h})^n$ and was used as the default shading model in standard graphics library such as OpenGL and Direct3D until recently.

Both Phong and Blinn-Phong models suffer from the same problems and are physically not plausible while only representing isotropic surfaces. For anisotropic surfaces an early model was proposed by Poulin and Fournier [1990] followed by the Ward reflection model (Ward [1992]) which was especially designed to fit measured BRDFs. The Ward model is able to represent both isotropic and anisotropic materials with a modification to the isotropic model to accommodate specular reflection in two principal directions. Later Ashikhmin and Shirley [2000] proposed an extension of the Phong model to represent both diffuse and specular reflectance exhibited by anisotropic materials.

The Cook-Torrance model (Cook and Torrance [1982]) is perhaps the most widely accepted BRDF model in the industry today. Being a physically-based model, it is able to represent both diffuse and specular surfaces with highlights, distinguishes between metals and dielectrics, uses the microfacet theory to describe rough surfaces and uses half angle to compute the specular lobe. It is given by

$$f_r(\omega_i, \omega_r) = \frac{D(\mathbf{h})F(\theta_r)G(\omega_i, \omega_r)}{\pi cos(\theta_r)cos(\theta_i)} \tag{2.3}$$

where $D(\mathbf{h})$ denotes the distribution function of the microfacet geometry, F denotes the fresnel reflection term, and G denotes the attenuation term due to shadowing and masking. The Cook-Torrance model is able to model a large variety of materials such as metals and plastics with rough surfaces and view-dependent effects. Walter et al. [2007] extended the Cook-Torrance model to better describe the microfacet distribution function D based on the work by Smith [1967]. The resulting model is termed as GGX model and is one of the most widely used physically-based model used in the CG industry today.

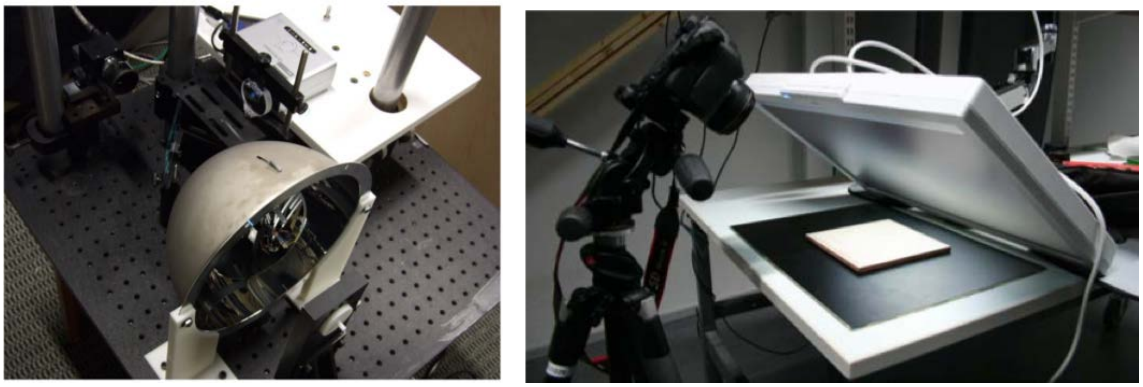


Figure 2.12: Material capture usually requires specialized hardware setup to capture high quality BRDF data such as a hemispherical gantry of Ghosh et al. [2010] (left) or a projector-light setup using an LCD monitor screen of Aittala et al. [2013] (right).

In an earlier work, Oren and Nayar [1994] enhanced the Lambertian model for rough diffuse surfaces such as sand, concrete and cloth by using V-shaped microfacet cavities described by a spherical Gaussian distribution.

To recover realistic material parameters we use the physically-based Cook-Torrance model (Cook and Torrance [1982]) for the real world surfaces we capture. Instead of using the GGX distribution (Walter et al. [2007]), our method use the Cook-Torrance model with a Beckmann distribution which is a physically-based distribution derived from Gaussian random surfaces as implemented in Mitsuba renderer (Jakob [2010]).

Next we look at how real world materials are captured using photographs. Earlier methods traditionally used special hardware for material capture but recently neural networks has been demonstrated to be very effective in material acquisition too. We discuss a few methods that required specialized hardware or capture setups. Then we discuss the two domains of material capture which has made material acquisition pretty lightweight – optimization-based and learning-based methods for practical material estimation.

We refer the interested reader to surveys on material capture: Guarnera et al. [2016]; Garces et al. [2022] and inverse rendering: Kato et al. [2020]; Tewari et al. [2021] for more general discussions of these broad topics.

2.3.3 Traditional material capture

The first measurements of BRDF were done using gonioreflectometers (Baribeau et al. [2009]; Obein et al. [2005]; Holopainen et al. [2009]) which are devices that measure the spectral reflectance of surfaces and can measure both specular and diffuse reflectance depending on its setting. The design of gonioreflectometers was proposed by Nicodemus [1965] and was subsequently used in the development of material models by Torrance and Sparrow [1967]; Blinn [1977]. The process of building and acquiring reflectance data using gonioreflectometers is costly and the device itself can be quite cumbersome.

To reduce the cost of acquiring BRDF data, researchers turned to image-based measurement methods which require general-purpose hardware such as a camera sensor and lighting setup. One of the first image-based setups for acquiring high quality BRDF data using images was presented by Marschner [1998]; Marschner et al. [1999]. The setup was highly accurate and complete and was able to measure data for a large number of homogenous materials including human skin. A similar setup was proposed by Matusik et al. [2003a] and was used to measure 100 isotropic BRDF. The MERL-MIT dataset (Matusik et al. [2003b]) is one of the largest and most reliable measured database of materials and is still widely used in material representation research.

Other setups which eliminate the use of moving parts and use optics to measure BRDF data using both reflected and refracted light are termed catadioptric measurement setups and are highly efficient. The imaging setup of Ward [1992] was designed to capture anisotropic surfaces and lead to the development of Ward BRDF model. High quality BTF data was captured by Dana et al. [1999] whose setup consisted of a robotic arm which held and rotated sample while being observed under a halogen bulb with a Fresnel lens and a video camera.

In a more recent work, Ghosh et al. [2009, 2010] described a setup which did not need any moving parts. The setup consisted of a hemispherical gantry and the measurement was done over a continuous region with a specially designed orthonormal basis function for illumination. A more general non-moving low-cost setup was built by Aittala et al. [2013] in which an LCD display was used to capture BRDF data using Fourier basis functions (see Fig. 2.12). The most lightweight setup which has proven successful in capturing material reflectance involves a mobile phone camera with a co-located flash and was first proposed by Aittala et al. [2015]. A number of subsequent works use this capture setup

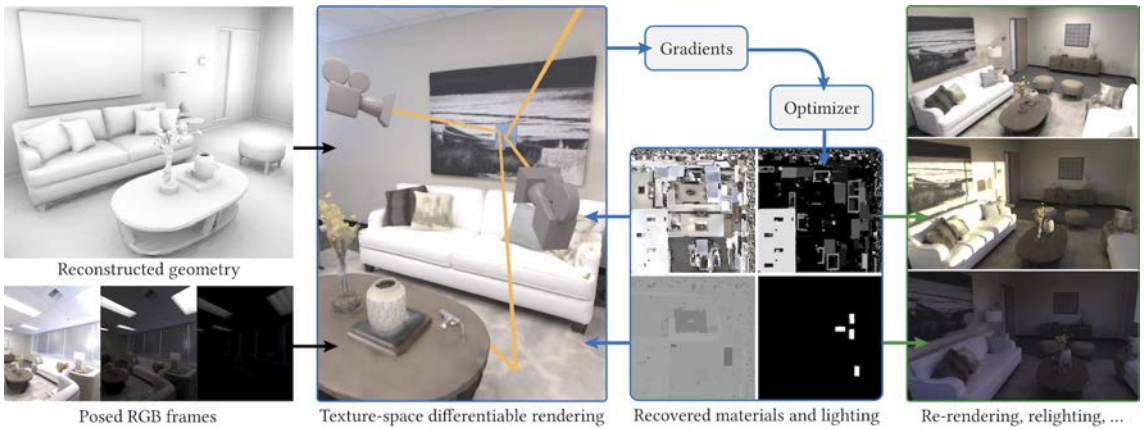


Figure 2.13: Nimier-David et al. [2021] proposed a texture-space differentiable rendering method to optimize and recover materials and lighting of a scene.

with deep learning to acquire materials quickly and easily.

2.3.4 Optimization-based material capture

The recent development of differentiable rendering algorithms has reinvigorated research on inverse rendering, as pioneered by Yu et al. [1999] for scene-scale material recovery. Given a 3D model of the scene, modern approaches estimate material and/or lighting by simulating complex global illumination effects using differentiable path-tracing. Using a differentiable Monte-Carlo renderer, Azinovic et al. [2019] recover material and illumination parameters over a synthetic scene using stochastic gradient descent optimization. While they show good results over synthetic scenes, the authors show limited use cases of their method for real captured scenes. Nimier-David et al. [2021] introduce a texture-space differentiable rendering method to optimize for materials and illumination for an entire scene (see Fig. 2.13). The optimization recovers good quality maps but needs up to 12 hours to do so on a single scene. Similarly, using a nested least-square optimization, Haefner et al. [2021] achieve reflectance recovery from HDR images in a scene setting but at great cost since the optimization runs per-object in scenes from the high qualtiy Replica dataset (Straub et al. [2019]).

Methods that also optimize for geometry have so far been limited to convex isolated objects often with specific lighting/capture constraints. Wu et al. [2016] take RGB-D Kinect images of an object under unknown lighting condition to jointly solve for camera pose, SVBRDF with specular lobe, lighting and normal by reconstructing the image. Using

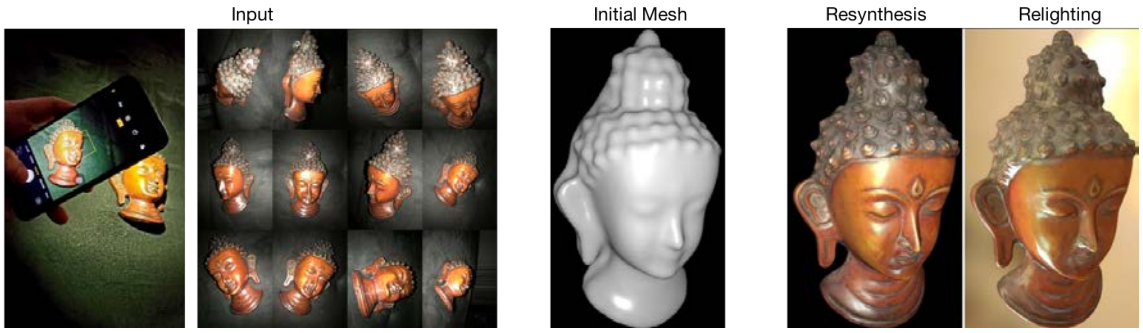


Figure 2.14: Luan et al. [2021] uses a set of ~ 50 images to optimize for the geometry, material and illumination of isolated objects enabling applications such as relighting and resynthesis.

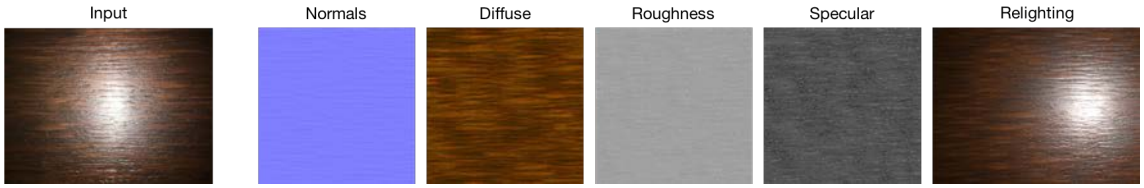


Figure 2.15: Capturing a photograph of a flat patch using mobile phone with co-located flash, methods such as Henzler et al. [2021] is able to recover the SVBRDF properties of the patch enabling relighting.

a multi-stage iterative joint optimization framework, Nam et al. [2018] provides a practical solution to capture 3D objects from a small set of images. Obtaining an initial estimate of geometry using SfM and MVS methods Goel et al. [2020]; Luan et al. [2021] use about 50 images to recover high quality reconstruction of shape and materials of captured objects (see Fig. 2.14). Recently Ma et al. [2021] proposed neural trace photography to learn non-planar high quality anisotropic appearance of an object by using free-form scanning via a hand-held mobile device consisting of a camera and LED array.

2.3.5 Learning-based material capture

Recovering material appearance from a few observations is an ill-posed problem for which machine learning offers practical solutions. By leveraging large datasets of images paired with ground truth material maps, deep convolutional networks can be trained to predict per-pixel material parameters. A line of work by Li et al. [2017, 2018a]; Deschaintre et al. [2018]; Gao et al. [2019]; Guo et al. [2021]; Zhou and Kalantari [2021]; Henzler et al. [2021] recovers SVBRDFs given a single picture of a flat surface patch captured with

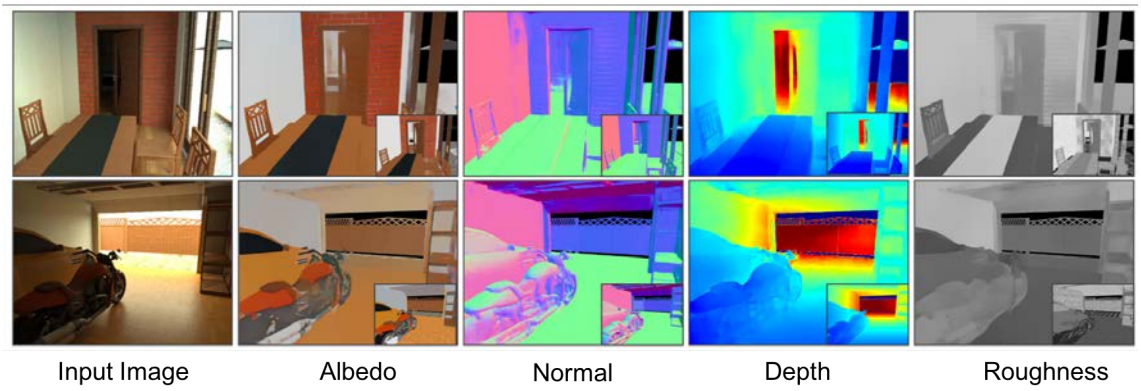


Figure 2.16: Inverse rendering using a single photograph to estimate the normals, depth, albedo, and roughness of the scene was proposed recently by Li et al. [2020].

a mobile phone camera using flash (see Fig. 2.15). The flash helps obtain visible cues which aid the network in identifying specular and glossy surfaces. Instead of recovering SVBRDFs, Rainer et al. [2019, 2020] proposed learning based solution to compress high-dimensional BTF data to render photorealistic materials.

Such methods were later extended to predict material, normal, and/or depth maps of isolated objects by Li et al. [2018b]; Boss et al. [2020]; Bi et al. [2020] and even of indoor scenes by Li et al. [2020] from a single image (see Fig. 2.16). Deschaintre et al. [2021] introduced a method using plane-polarized images to recover shape and SVBRDFs of objects. Most recently Li et al. [2022] proposed a solution for scene scale material acquistion which takes as input a single image of an indoor scene along with a 3D model of that scene, and build upon single view material prediction by Li et al. [2020] to assign procedural material models to object parts.

Material estimation is a hard problem to solve, especially with limited information from a single view which can leave the solution highly underconstrained. To account for this, researchers have tried using multiple views to estimate surface appearance properties. Aittala et al. [2015] was the first to propose use of two views of a patch of stationary textured materials to recover its SVBRDF parameters. Multi-view information needs to be properly aggregated to be fed to a neural network. Prior methods on multi-image material prediction such as Deschaintre et al. [2019]; Guo et al. [2020]; Asselin et al. [2020] only partially address these challenges since they focus on small planar patches and assume that each point is visible in all input images. Deschaintre et al. [2019] used

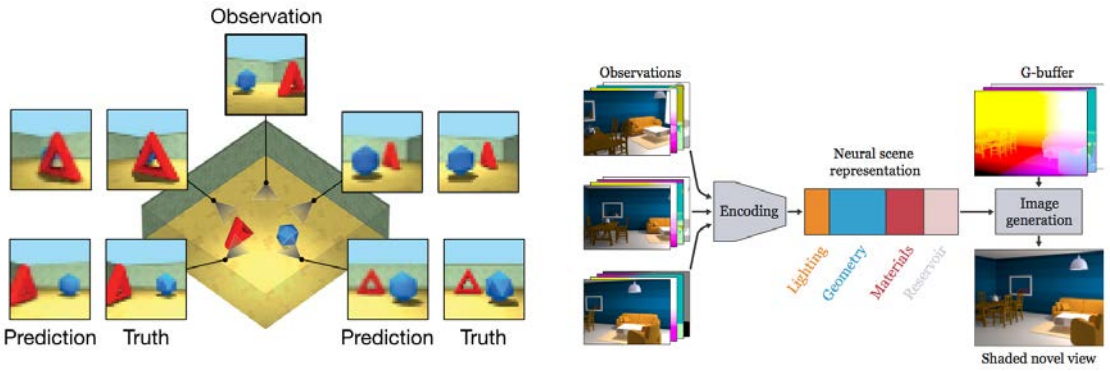


Figure 2.17: Learning a scene representation using a neural network to re-render them in different states was demonstrated by works such as Eslami et al. [2018] (left) and Granskog et al. [2020] (right).

one network per-image with shared network weights and a pooling layer to encode multi-view information. The multi-view images were used as a strong prior by Guo et al. [2020] to help find the right latent space representation in the learned space to best explain the appearance of the material samples. While Ye et al. [2021] describe how to combine image warping and max pooling to handle videos captured with a moving camera, they again focus on planar surfaces free of occlusion, and recover material parameters over a reference frame rather than over all input views of a 3D scene.

We tackle the much harder problem of *scene-scale* material recovery from multi-view observations of the scene. In this setting the solution is much harder to obtain because of complex global illumination effects such as shadows and inter-reflections as well as occlusions due to geometry. In our solution we use a learning-based approach instead of optimization since we do not know or estimate illumination and thus can not attempt to reconstruct the input images as done by other works. The advantage of a learning-based approach is that it is fast and generalizable over multiple scenes.

2.4 Neural Scene Representations

Earlier we discussed Mildenhall et al. [2020] in the context of neural rendering for novel-view synthesis. This is one amongst a line of works which attempt to learn the scene with a neural representation. In an early work before the general adoption of neural networks, Ren et al. [2013] attempted the difficult task of learning indirect illumination of a scene with varying roughness with neural networks. The network was trained on a

set of pre-rendered synthetic datasets. Later Nalbach et al. [2017] proposed a network to learn screen-space shading to simulate global illumination in images. They provided the network with a set of attributes from deferred shading buffers such as position, normals, and reflectance to learn screen-space shading effects, which is simulated at runtime by the network to create arbitrary combinations of these effects with good quality and speed.

More recent works include Eslami et al. [2018] that learns a variable scene by showing the network images of the scene in different configurations. It was thus able to extrapolate on the learnt variables to render the scene from novel viewpoints. This work was extended by Granskog et al. [2020] which introduced auxiliary buffers to aid the network with the learning task along with having a structured neural scene representation (see Fig. 2.17). Diolatzis et al. [2022] builds upon this work to train a network for variable scene representation by online generation of data samples with an *active exploration* scheme which helps the network learn difficult scene configurations more efficiently. For static synthetic scenes with variable lighting, Müller et al. [2021]; Rainer et al. [2022] provide neural network based solutions for learning global illumination of the scene to render them in real-time.

NeRF (Mildenhall et al. [2020]) has been shown to be highly effective for novel-view synthesis of real data. There has been a number of works which query NeRF to recover the surface and material appearance of an object. Most prominent of these are Srinivasan et al. [2021]; Zhang et al. [2021b]; Boss et al. [2021a,b] which help decompose the scene geometry, appearance and illumination to efficiently relight an object. Zhang et al. [2021a] works on objects as well but recovers the surface in the form of SDFs instead of triangular meshes along with high quality material and illumination, and demonstrates editing the object's material and lighting. In another recent work Munkberg et al. [2021] combine neural and traditional representations within a differentiable rendering framework to recover a triangle mesh, an SVBRDF texture and an environment map of isolated objects.

For the task of relighting, a number of works do not recover traditional material parameters. Philip et al. [2021] feeds multi-view information to a neural renderer to perform novel-view synthesis with relighting without obtaining any intermediate appearance parameter. Extending the variability from relighting and novel-view synthesis, Bemana et al. [2020] propose learning *X-fields* which support varying time as another dimension for a real scene.

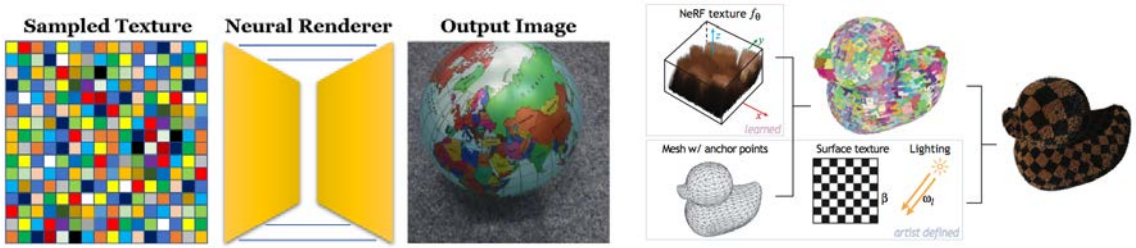


Figure 2.18: *Neural textures* have been used by methods such as Thies et al. [2019] (left) to enable plausible relighting and in some cases editing of materials such as Baatz et al. [2021].

Some works such as Thies et al. [2019]; Gao et al. [2020]; Meka et al. [2020]; Baatz et al. [2021] learn *neural textures* to perform high-quality novel-view synthesis with specular highlights and relighting using structured or unstructured capture (see Fig. 2.18). While very effective for overall scene relighting, editing of these neural textures to achieve material editing such as change in surface roughness or underlying albedo have not been demonstrated efficiently. Most neural textures are also not compatible with traditional image synthesis pipeline since the textures are not interpretable and cannot be used to render using a path tracer. While NeRF-Tex (Baatz et al. [2021]) allows editing of the material and is used with a volumetric path tracer to render, it is only defined for very fine-scale mesoscopic materials such as grass and fur and not for widespread macroscopic materials like wood, leather, paint, metals etc.

We use a learning-based method to recover explicit material parameters of the physically-based Cook-Torrance model. Recovering interpretable parameters helps the user to edit the material as desired. The recovered material maps are also compatible with traditional photogrammetry pipelines and are used to re-render the scene with a path tracer. Moreover we study rendering a scene using neural networks with interpretable material models as done by Diolatzis et al. [2022]. We use the learnt neural scene representation to render synthetic images with variations to create a dataset for speeding up training and data generation process. We study the pros and cons of using neural rendering as an alternative to traditional rendering for data generation.

Chapter 3

Hybrid Image-based Rendering

Image-based Rendering (IBR) algorithms present an interesting alternative over traditional rendering for two main reasons. First, it is much easier to create assets in a 3D scene, and second IBR allows much faster free-viewpoint navigation by avoiding the need for costly global illumination computation. IBR provides a rich toolset for free-viewpoint navigation in captured scenes. Many IBR methods exist, usually with an emphasis either on image quality or rendering speed. In this chapter we identify common IBR artifacts and combine the strengths of different algorithms to strike a good balance in the speed/quality tradeoff.

In the process of rendering a scene efficiently with IBR, we solve several significant subproblems. First, we address the problem of visible color seams that arise from blending casually-captured input images by explicitly identifying view-dependent regions and preserving their effects. Second, we compensate for geometric reconstruction errors by refining per-view information using a novel clustering and filtering approach. Finally, using a rough estimation of the underlying surface appearance, we introduce a hybrid IBR algorithm which treats view-dependent regions differently as opposed to regions which are mostly diffuse. Our algorithm locally identifies and utilizes the rendering method best suited for an image region while retaining interactive rendering rates. We compare our method against classical and modern (neural) approaches in indoor and outdoor scenes and demonstrate superiority in quality and/or speed.

3.1 Introduction

Freely and *interactively* navigating in virtual representations of captured 3D scenes has widespread applications, e.g., urban planning, training in virtual reality, games or augmented reality. Recent Structure-from-Motion (SfM) (Snavely et al. [2006]) and Multi-View Stereo (MVS) (Goesele et al. [2007]) solutions allow the creation of a textured 3D mesh of a scene with only photos or video as input. However, these meshes have

only diffuse appearance, and often have geometric reconstruction errors, degrading realism. Image-Based Rendering (IBR) (Shum et al. [2008]) and recent neural rendering methods (Tewari et al. [2020]) address some of these issues, using multiple photos to compensate for geometric errors and to render view-dependent effects such as glossy appearance. However, current methods still suffer from visual artifacts and can be slow.

We target a new IBR algorithm that addresses previous artifacts, focusing on scenes with wide-baseline capture, while allowing free-viewpoint navigation far from the input views at *interactive frame rates*. The first artifact we consider occurs when blending multiple photos of a given scene for IBR: each photo can be captured with different camera parameters (exposure, white balance, etc.) or lighting may change during capture. As a result, multiple views of a given diffuse surface do not blend well in a novel IBR view, creating visible seams. Previous methods fail to satisfactorily harmonize our wide-baseline datasets (HaCohen et al. [2013]; Huang et al. [2017]), and are not designed to preserve view-dependent effects, such as glossy material appearance. We want to *identify these view-dependent regions and preserve them* in the images blended during IBR. The second artifact is related to geometric errors. SfM/MVS generate a *global mesh*. The fusion step of MVS is based on photoconsistency among views. Many hard cases such as non-diffuse surfaces, small or thin structures etc., are not well reconstructed resulting in visual artifacts for IBR methods using the global mesh (Buehler et al. [2001]; Eisemann et al. [2008]). One effective solution is to use *per-view* information (Chaurasia et al. [2013]; Hedman et al. [2018]) which is not always multi-view consistent, but can capture missing geometric details. This information is valid close to input views, but can result in artifacts in distant novel views, depending on the blending strategy.

Finally, there is a tradeoff between rendering speed and image quality in all algorithms used to display multi-view datasets. The textured mesh is much faster than IBR, but is only valid in diffuse, well reconstructed regions. IBR algorithms using the global mesh geometry can reproduce glossiness, but suffer in badly reconstructed regions (ghosting, blurring), where per-view solutions are better. To provide the best speed/quality tradeoff, we need to identify each case, and efficiently combine the advantages of each method.

Our method addresses these issues. First, we propose a new color harmonization approach, where we use a textured mesh as a view-independent basis for diffuse harmonization, and *photometric variance* to identify view-dependent regions. We then re-inject these

regions into the input images using techniques akin to digital photomontage (Agarwala et al. [2004]). This results in modified input images with diffuse regions that can be seamlessly blended in IBR while *preserving view-dependent* content. Second, we analyze the depth errors of view-dependent meshes (Hedman et al. [2018]), and propose a new filtering approach based on clustering of depth of the different meshes, followed by spatial filtering. Finally, we propose a new *hybrid IBR* method by analyzing the strengths and weaknesses of different rendering algorithms. We identify regions where the global mesh is sufficient by estimating geometric uncertainty and use our filtering-based per-view mesh IBR algorithm for uncertain regions. For regions where the global mesh is of good quality, in glossy regions we use a rough estimate of underlying surface appearance to render with ULR (Buehler et al. [2001]) else we use a textured mesh for diffuse surfaces. Our method provides a good quality/speed tradeoff, even compared to deep-learning based solutions such as Deep Blending (Hedman et al. [2018]).

In summary, we present three main contributions:

- A new multi-view image harmonization algorithm that removes color differences in diffuse regions and *re-injects view-dependent effects* such as glossy appearance in the harmonized input images.
- A new blending algorithm for per-view meshes based on depth clustering and spatial filtering.
- A new, efficient hybrid IBR method based on a rough estimate of underlying surface appearance that strikes a good balance in the speed/quality tradeoff.

We show results on several indoors and outdoors scenes, and provide comparisons with previous IBR and neural rendering algorithms, clearly illustrating the speed and/or quality superiority of our method. Our entire *hybrid IBR* pipeline is summarized in Fig. 3.1. The source code, supplemental materials and datasets can be found here: `https://repo-sam.inria.fr/fungraph/hybrid-ibr/`

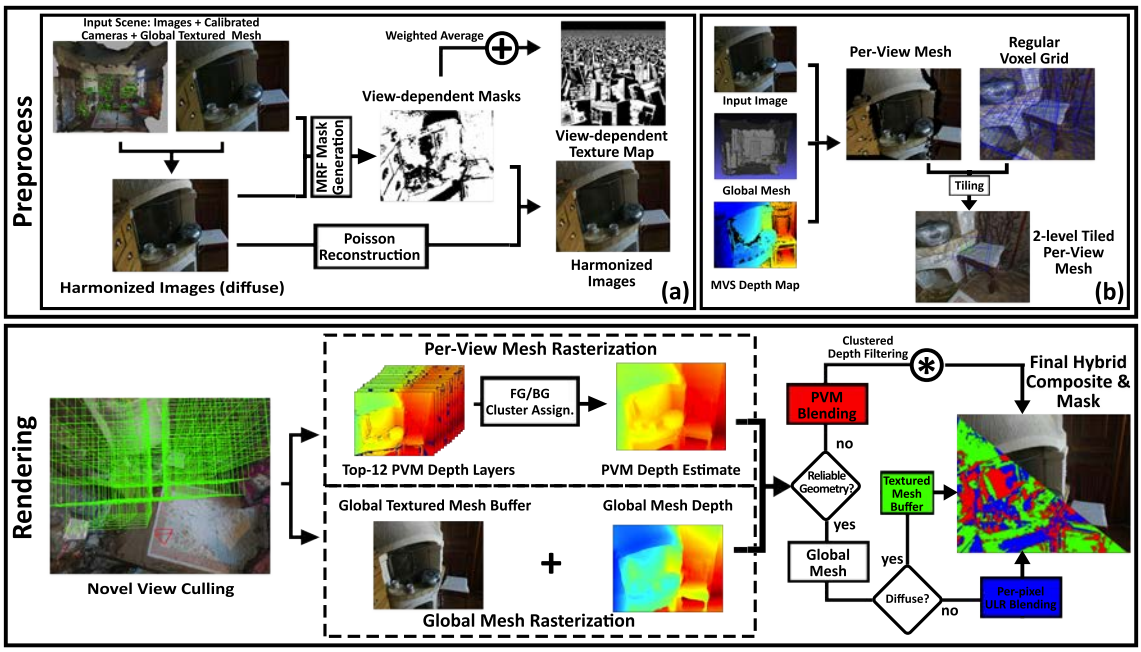


Figure 3.1: The first step of our method takes an IBR scene and reduces color seam artifacts by harmonizing input images in a pre-processing step to obtain harmonized images *(a)*. Next, we compute per-view meshes (PVMs) similar to Hedman et al. [2018] resulting in our 2-level tiled PVMs *(b)*. For rendering, we start by rasterizing the global mesh and PVMs to get a global and per-view depth estimate. We compare the global and per-view depth maps to decide if PVM rendering should be used instead of the global mesh. Finally, when rendering with the global mesh, we check if a pixel is diffuse using the view-dependent texture map obtained in *(a)* and use the textured mesh buffer for the final color, else we perform per-pixel ULR blending and use the blended color.

3.2 Related Work

We discuss previous work on harmonizing images for blending and stitching applications, related to our harmonization solution. Then we give an overview of the most closely related image-based and neural rendering algorithms as discussed in Chapter 2 and highlight how we improve upon these previous solutions.

3.2.1 Image Harmonization

Seamless blending is a long-standing problem in computer vision and graphics. Many applications such as panoramic image stitching, texture synthesis, photogrammetry along with image-based rendering require seamless blending.

Implicit methods such as Kim and Pollefeys [2008] and Goldman [2010] were proposed to prevent seam artifacts due to unknown vignetting, exposure and camera response functions causing appearance variation in multi-image datasets. These methods work best for 2D image operations on images captured with specific constraints but can be difficult to adapt to multi-view datasets. Zhang et al. [2016] propose a solution for 3D datasets which recovers per-image exposure and per-vertex radiance of the mesh to radiometrically calibrate the scene. Color transfer methods have also been used to harmonize images. Moulon et al. [2017] use common color segments between a reference and a target image and solve a linear optimization to recover gain/bias factors per image handling shutter speed and aperture variations. Transfer of image color statistics using a reference image was also explored in HaCohen et al. [2011] using an extension of the PatchMatch algorithm to find Non-Rigid Dense Correspondence (NRDC) regions and then fit a parametric color model for global color transfer from a reference image to the target image. This method was extended by HaCohen et al. [2013] for multi-image datasets. While these methods provide good color harmonization, they fail on large datasets such as ours, which have insufficient overlapping regions between distant images and in general are not designed to correctly handle view-dependent effects.

Multi-view mesh texturing solutions implicitly address the problem of color harmonization. Lempitsky and Ivanov [2007] use a Markov Random Field (MRF) (Kwatra et al. [2003]) that incorporates a pairwise penalty for color difference at seams of model polygons. Follow-up work (Zhou and Koltun [2014]; Waechter et al. [2014]) extends this by formulating an optimization problem having a data term for view selection and smoothness term for color correction. Recently, Huang et al. [2017] used the NRDC color model to obtain seamless textures by solving an optimization to recover a parametric curve per image and adjusting the intensities accordingly. Commercial solutions (Reality [2018]; Jancosek and Pajdla [2011]) produce diffuse textures with similar properties. We use a harmonized textured mesh as a basis for our solution, but we *preserve* pixels in the input images corresponding to *view-dependent effects*.

3.2.2 Image-Based Rendering

As we discussed in Sec. 2.1.2, rendering a scene from captured images has a long history in computer graphics. From early works that explored view interpolation (Chen and Williams [1993]; Chen [1995]; McMillan and Bishop [1995]), IBR evolved to use different

scene models computed from the captured photographs such as light fields (Levoy and Hanrahan [1996]), Lumigraph (Gortler et al. [1996]) and eventually mesh-based rendering (Debevec et al. [1996, 1998]). Unstructured Lumigraph Rendering (ULR) (Buehler et al. [2001]) allowed IBR for casual capture by blending input images based on a geometric proxy. With advances in structure-from motion (SfM) and multi-view stereo (MVS), using explicit geometry has become a viable option for IBR. In this thesis, we call the geometry reconstructed by MVS the *global mesh*.

Further improvements focused on rectifying the global mesh to model floating geometry (Eisemann et al. [2008]) or using ambient point clouds for view interpolation (Goesele et al. [2010]). But the global mesh still suffered with poor reconstruction. Alternative solutions emerged in the form of *per-view* solutions such as the method by Chaurasia et al. [2013] and InsideOut (Hedman et al. [2016]). In both cases, geometric details incorrectly reconstructed or even missing in the global mesh are corrected. We build on Inside Out per-view meshes to propose a new approach which handles uncertainty in local depth when seen from novel views. Reflections are a special case of view-dependent effects, that have been explicitly handled in previous work (Sinha et al. [2012]; Kopf et al. [2013]; Rodriguez et al. [2020b]) using layered rendering or semantics of scene content. Our hybrid algorithm takes inspiration from these ideas and explicitly computes view-dependent regions using photometric variance, using it as a rough estimate of the underlying surface appearance, to treat them differently from view-independent regions.

Ortiz-Cayon et al. [2015] presented an approach to determine whether a simple homography can selectively replace the more expensive warps of the superpixel-based solution of Chaurasia et al. [2013] in a pre-process. Our hybrid rendering algorithm is more powerful such that it selects the rendering algorithm for each pixel and each novel view.

Deep Blending (Hedman et al. [2018]) improved the per-view geometry of Inside Out by combining high-quality depth maps (Schönberger et al. [2016]) with a more complete (but possibly inaccurate) global mesh (Reality [2018]; Jancosek and Pajdla [2011]). In addition, depth discontinuities and mesh edges are treated carefully, resulting in higher quality meshes compared to Inside Out. The method then learns blend weights to correct remaining artifacts, and, notably, to get rid of erroneous and floating geometry.

Deep Blending is one of the early *neural rendering* (Tewari et al. [2020]) solutions. Thies et al. [2018] separates diffuse and specular components of a scene and learns the view-

dependent effects explicitly. Recently, Riegler and Koltun [2020] use a recurrent neural network to learn warping and blending input views to novel views separately; we compare to this method in Sec. 3.7.

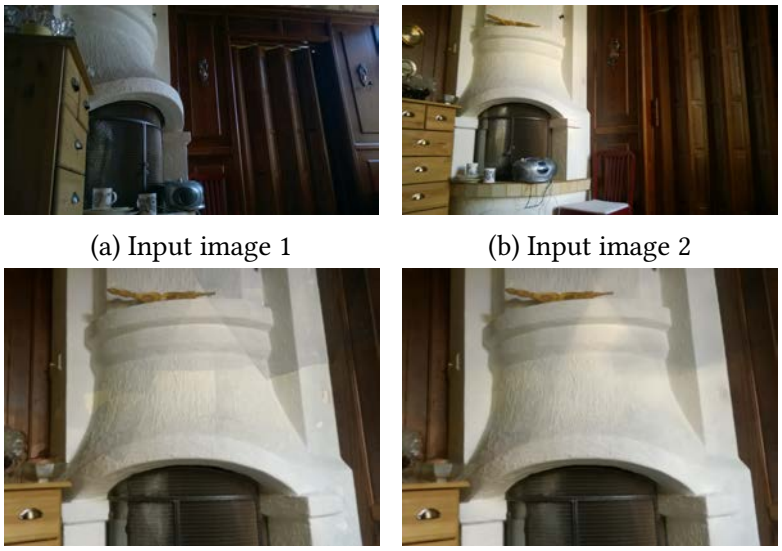
Other neural rendering methods aim to synthesize novel views by learning implicit scene representations (Mildenhall et al. [2019]; Thies et al. [2019]; Flynn et al. [2019]; Meshry et al. [2019]) or explicit volumetric representations (Zhou et al. [2018]; Sitzmann et al. [2019]; Srinivasan et al. [2019]; Lombardi et al. [2019]; Mildenhall et al. [2020]; Martin-Brualla et al. [2021]). Most of these methods focus either on capturing a single object, or require small-baseline capture that allows only small motion around input views. Importantly performance of most of these methods is typically in (tens of) seconds or even minutes per frame, precluding their use for *interactive* applications. As reported by (Koltun [2020]), the typical runtime for novel-view synthesis using Neural Radiance Fields (NeRF) (Mildenhall et al. [2020]; Martin-Brualla et al. [2021]) is 1-2 minutes and roughly 1 second using Riegler and Koltun [2020]. Our method takes around 30 ms on average and can provide frame-rates as high as 50Hz (20 ms).

Our solution, based on improved image harmonization, spatial filtering and a hybrid rendering algorithm which takes advantage of the underlying surface appearance, avoids the use of a neural network altogether, achieving a good balance between speed and quality. Our hybrid renderer runs at much faster frame rates than all current neural rendering algorithms, allowing *interactive* free-viewpoint navigation of captured scenes.

3.3 Analysis and Motivation

As discussed in the introduction, we address three major issues for wide-baseline IBR: 1. Eliminating visual artifacts due to differences in diffuse surface appearance due to capture errors, while *preserving* desirable view-dependent effects. 2. Removing residual artifacts due to geometric reconstruction errors using per-view geometry. 3. Dealing with the quality-speed tradeoff by carefully selecting between different algorithms.

We will analyze each issue below, including the strengths and weaknesses of different IBR algorithms, helping us develop our new hybrid solution. In particular, we consider Inside Out (Hedman et al. [2016]) and Deep Blending (Hedman et al. [2018]) that use per-view meshes (PVM) to compensate for geometric inaccuracies. In addition to the deep-learning approach to blending, Hedman et al. [2018] introduced a baseline heuristic



(a) Input image 1 (b) Input image 2

(c) PVM-H (d) Deep Blending

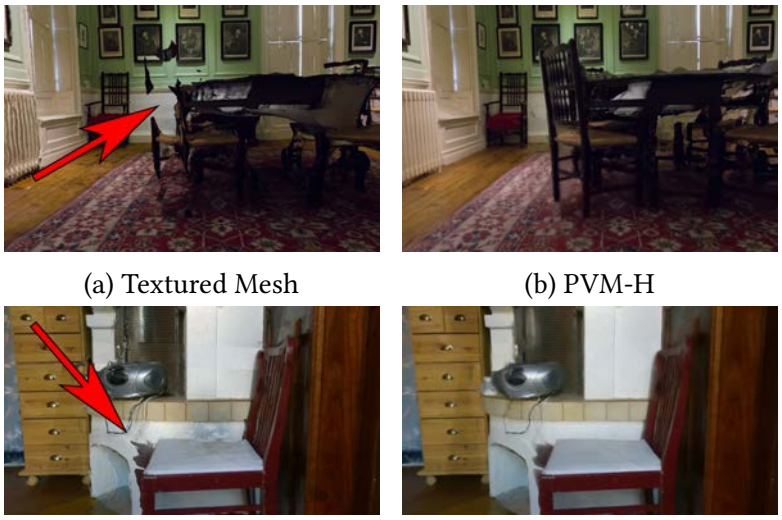
Figure 3.2: Top: The chimney has different colors due to exposure differences in the two images. The color differences on the metal handles should be preserved. Bottom left: Novel view with PVM-H has significant seams. Bottom right: Deep Blending reduces the seams, but residual artifacts remain.

method to blend PVMs without a neural network; we refer to this method as PVM-H from now on.

3.3.1 Harmonization with View-Dependent Effects

When capturing multi-view datasets there can be subtle changes in illumination due to internal settings of the camera – even when some exposure settings are fixed – or changes in lighting during the capture session. These differences in input images show up as discontinuities or *seams* in novel views during IBR, when blending pixels from different input images. Note that learned weights in the recent Deep Blending (Hedman et al. [2018]) algorithm reduce such seams, but residual artifacts remain, as shown in Fig. 3.2. Previous methods (Sec. 3.2) remove the view-dependent effects we want to preserve.

Our goal is to achieve results on par or even better than Deep Blending in terms of removing seams. We do this by *harmonization of diffuse* surfaces while explicitly *identifying and preserving* view-dependent effects.(Sec. 3.4).



(a) Textured Mesh (b) PVM-H

(c) PVM-H (d) Deep Blending

Figure 3.3: Top left: The textured mesh is missing the chair. Top right: Per-view meshes are able to reconstruct the missing detail. Bottom: An extreme case (left), where a PVM creates visual artifacts, that even the learned weights of Deep Blending (right) cannot correct.

3.3.2 Compensating for Geometric Reconstruction Errors

Early image-based rendering algorithms used the global mesh which suffers from reconstruction errors, typically due to the uncertainty in the depth maps computed by multi-view stereo algorithms, and in the subsequent fusion step. PVM-based methods (Hedman et al. [2016, 2018]) reduce artifacts due to such errors. This is clearly illustrated in Fig. 3.3, where the chair is missing from the global textured mesh (Fig. 3.3a), but present in the rendering using PVM-H (Fig. 3.3b).

PVMs can correct erroneous geometric reconstructions in the global mesh, but are not necessarily multi-view consistent. As a result, when moving away from the corresponding input view, PVMs can introduce artifacts. An extreme case is shown in Fig. 3.3c, where a PVM from the front of the chair extends incorrectly. Deep Blending (Fig. 3.3d) tries to resolve these issues by assigning a low blend weight to PVMs responsible for artifacts. However, in such cases, the weighting strategy is insufficient. Resolving such uncertainty in depth by improving the pre-processing step for PVMs (Hedman et al. [2018]) is hard because the input images do not capture the required depth information in any view.

To summarize, PVMs can correct for missing and erroneous geometry of the global mesh,

Figure 3.4: Textured mesh preserves sharp details in diffuse regions that are lost due to blending in ULR and PVM-H (blue inset). ULR corrects minor geometric errors in textured mesh typically in glossy regions (yellow inset). Large geometric errors, such as ones due to occlusion edges, are rectified by per-view geometry improving PVM-H rendering quality over ULR (red inset).

but can also result in artifacts far from the input views. These observations guide our approach to improve per-view mesh IBR by *estimating uncertainty* and using a novel filtering method for view synthesis.

3.3.3 Quality-Speed Tradeoff for IBR

We consider three algorithms for displaying multi-view content: the textured mesh (TM), produced by any MVS system, per-pixel unstructured lumigraph rendering (ULR) (Buehler et al. [2001]) and finally per-view mesh rendering (PVM). The discussion here of PVM is valid both for the PVM-H approach of Hedman et al. [2018] and our improved solution (Sec. 3.5). We analyze the tradeoffs of these algorithms, motivating our hybrid rendering solution (Sec. 3.6); we start with a cost breakdown of different steps for each algorithm (Table 3.1).

Table 3.1: Average cost of each step for the different rendering algorithms (see Sec. 3.5 for details).

Steps/Algorithm	TM	ULR	PVM-H	DB
Global Geometry	0.5 ms	0.5 ms	0.5 ms	0.5 ms
Voxel Lookup	-	-	3.4 ms	3.1 ms
Tiles Sorting	-	-	1.3 ms	1.5 ms
PV Depth Pass	-	-	0.6 ms	0.6 ms
Blend Cost Compute	-	-	3.6 ms	-
Mosaic Generate	-	-	-	5.4 ms
Blending	-	4.5 ms	0.2 ms	51.6 ms
Total	0.5 ms	5.0 ms	9.6 ms	62.7 ms

Textured Mesh (TM)

Rendering a scene with TM incurs negligible cost on modern GPUs using a single fetch operation per fragment. The other two algorithms also rasterize a g-Buffer from the global mesh: ULR uses the depth per fragment to lookup color in input images, and PVM uses the depth to determine visible PVMs; we can thus render TM at every frame at no extra cost. Novel views are very stable with TM, but the rendered images can lack realism in regions with poor reconstruction, and for non-diffuse objects. On the positive side, TM – unlike all IBR methods – does not blend multiple input images which can avoid blur in some cases.

Unstructured Lumigraph (ULR)

The original ULR algorithm (Buehler et al. [2001]) triangulates the image and blends images over this image-space mesh. With high-quality MVS meshes, we use deferred shading where ULR weights are applied at each visible point of the global mesh to blend a subset of input images (we preselect the 4 best using ULR weights and then blend these). This algorithm was used as a baseline comparison in several previous IBR solutions such as Hedman et al. [2018]; Rodriguez et al. [2020b]; Mildenhall et al. [2019].

Per-pixel ULR preserves view-dependent effects, since for each pixel in the novel view, the algorithm re-projects and blends colors from a subset of input cameras, i.e., a highlight moves from its position in one input view to another. It also reduces artifacts due to minor reconstruction errors in TM as shown in Fig. 3.4 (yellow inset). Interestingly, this happens in regions which exhibit glossiness that in turn introduces error in reconstruction. Thus rendering these pixels with ULR over TM improves quality both in terms of reconstruction error and view-dependent appearance. In addition, we see in Table 3.1 that this version of ULR is faster than PVM.

ULR can also treat distant backgrounds, where MVS systems often create large textureless triangles. As previously noted by Rodriguez et al. [2020b], PVMs are unsuitable for such regions, because they are sparsely covered in input images and would be wasteful.

Per-View Meshes (PVM)

As discussed in Sec. 3.3.2, PVMs improve image quality by correcting erroneous geometry of the global mesh, but can result in significant artifacts far from input views. We make

the empirical observation that for a given novel view, a majority of PVMs provide the correct depth. This suggests a new depth filtering strategy that can reduce some of the artifacts due to PVMs. From Table 3.1 we see that PVMs are slower than ULR but are still 6 times faster than the full deep-learning pipeline.

These observations guide the design of our hybrid IBR algorithm (Sec. 3.6 and Fig. 3.6).

3.4 Image Harmonization

In wide baseline multi-view datasets, images may have significant appearance variation between views for the same region, (see Fig. 3.5a), as explained in Sec. 3.3.1. We will utilize multi-view consistency to harmonize appearance while explicitly preserving the view-dependent effects, in two steps. We give an overview of the steps involved in Fig. 4.1a. First we harmonize the input images as if they were entirely diffuse, using the textured mesh as a baseline for color correction. Next, we explicitly identify view-dependent regions that we want to preserve, and re-inject them into the harmonized images similar to digital photomontage (Agarwala et al. [2004]). The final processed images are correctly harmonized in diffuse regions, while preserving desirable view-dependent effects such as glossy or specular surfaces (see Fig. 3.5).

3.4.1 Diffuse Harmonization

We use the textured mesh as a baseline for color correction, since it provides a stable common reference point for appearance across input views without view-dependent effects. The global mesh texture provides a base color for each vertex and is typically free from incorrect color seams, serving as a good baseline for image harmonization. We experimented with different texturing methods, e.g., Waechter et al. [2014] and a simple approach using ULR weights to blend textures provided in Bonopera et al. [2020]. We obtained best results using the textured mesh produced by RealityCapture (Reality [2018]), which we use for all our tests. It should be noted that our solution can work with Delaunay tetrahedralization meshes reconstructed using any off-the-shelf solution such as COLMAP (Schönberger et al. [2016]) and textured using any open-source texturing method.

We project each vertex in the cameras from which it is visible, compute the per-channel standard deviation of color intensities corresponding to each pixel associated with the

vertex and store it as the vertex color, in the *variance mesh*.

Recovering a single exposure factor per-image, as done in previous work (Huang et al. [2017]), does not capture color variations due to changes in lighting in localized image regions. Instead we compute a per-pixel, per-channel *scaling factor* for each input image, correcting each pixel to be close to the corresponding texture value, and applying a Gaussian kernel for smoothness. We perform the following operation to obtain the output pixel value I^{lin}_{out}

$$I^{\text{lin}}_{out} = \frac{G * T^{\text{lin}}}{G * I^{\text{lin}}} I^{\text{lin}},$$

where G denotes a Gaussian kernel (filter size 3% of dominant image resolution), $*$ denotes convolution, T^{lin} is the texture, I^{lin} the linearized input image and all operations are applied independently for each color channel, and in linear color space to ensure meaningful image transformations, by inverting gamma correction.

We leave out saturated pixels with an intensity in the highest or lowest 2% range from the computation. Additionally, we leave out pixels with color variance greater than 10% of the color range, which worked best empirically, identified by back-projecting the variance mesh to input images.

3.4.2 Re-injecting View-Dependent Pixels

Input images are now harmonized, removing seams when blending for novel view synthesis. However, desirable view-dependent effects such as glossy highlights are also lost. We re-inject this information in two steps. First, we identify regions with view-dependent effects by computing a harmonization mask for each input image. Second, for each view-dependent region in each input image, we re-introduce the original image statistics.

To compute the harmonization mask we start by formulating a graph-cut based energy minimization problem (Kwatra et al. [2003]) to recover a *binary mask*. The mask indicates if the pixel should be changed to contain view-dependent information or remain unchanged in the harmonized image. We use a method similar to (Agarwala et al. [2004]); details are provided in Appendix A.1. The mask obtained from the MRF is combined (binary "and") with the regions which were pre-identified as saturated and highly view-dependent in Sec. 3.4.1.

Once the regions are identified, we perform Poisson blending (Pérez et al. [2003]) to avoid

(a) Input (b) Mask (c) Harmonized

Figure 3.5: Harmonization: From an input image (a) exhibiting significant appearance variation, we obtain harmonized images (c). A mask (b) alleviates specularity suppression.

seams on region boundaries. We use diffuse harmonized images as a guide to re-introduce the gradients of the original image in the regions identified (in black) using the binary mask (see Fig. 3.5b). We see that shiny materials (fireplace protection, cassette player, drawer handles) are marked as view-dependent. The final harmonized image for a given pair of images is shown in Fig. 3.5c.

3.5 Per-view Mesh-based Rendering

In this section we address artifacts due to PVM rendering that typically occur far from the input views (Sec. 3.3.2), by introducing a two-level voxel structure and a spatial filtering step.

3.5.1 Storage and Acceleration

Previous methods (Hedman et al. [2016, 2018]) store PVMs as *tiles* in a voxel structure, allowing fast rendering and camera selection for blending. PVMs are "sliced" in voxels, based on spatial location; the slice and voxel are called a *tile*. Deep Blending (Hedman et al. [2018]) tries to address depth ambiguity using learned weights. Instead, we identify

erroneous input and either blend with a low weight or discard it altogether.

Our initial observation is that the granularity of the original tiles is too large for an effective elimination strategy. As a consequence, we introduce a two-level hierarchy, providing finer granularity: We use a coarse grid of resolution 32^3 to fetch visible voxels in a novel view, and a fine grid to slice the per-view meshes. The fine grid subdivides each coarse grid cell using a sub-sampling factor of 4 per dimension. The process of generating per-view meshes is summarized in Fig. 3.1b. For a detailed explanation of the steps involved, we refer to Section 4 from Hedman et al. [2016].

Our PVM rendering starts by rasterizing the global mesh, and marking all voxels intersected as visible. For each visible voxel at the fine scale we rank the input tiles to obtain a sorted list of the top-12 input tiles per voxel similar to Hedman et al. [2016]. We then rasterize all 12 tiles in parallel using indirect layered rendering.

3.5.2 Clustered Depth Filtering

Most rasterized tiles tend to have a correct depth, but occasionally outliers occur. We address this problem building on the assumption that a majority of tile depths is in fact correct. The intuition is to separate the foreground from the background since the PVMs contain either correct previously unreconstructed geometry (see Fig. 3.3b), or exhibit incorrect "over-reconstructed" geometry (see Fig. 3.3c).

First, we find two depth estimates per pixel using k-means clustering - the potential foreground and background. We obtain a blended color $c_{i,k}$ per pixel i and cluster k, by calculating ULR weights (Buehler et al. [2001]) for the candidate colors within the cluster. These assign higher weight to views that are closer to the novel view in position and viewing direction.

We now need to determine how to weigh the two clusters for rendering. In areas with significant geometric reconstruction errors, clustering may be inconsistent across neighboring pixels. We address this by introducing a spatial filter over the depth candidates to keep depth in uncertain regions locally consistent. Intuitively, we encourage pixels to maintain the same depth in a local neighborhood, weighted by proximity. Given a novel view with two depth candidates k per pixel i, we estimate weights $W_{i,k}$ taking the spatial

neighborhood Ω_i around pixel i into account

$$W_{i,k} = \sum_{j \in \Omega_i} \sum_{k'} G_{i,j;\sigma} \exp(-\alpha(d_{j,k'} - d_{i,k})) \# N_{k'}. \tag{3.1}$$

Here, $\# N_k$ is the number of depth candidates for cluster k, and $d_{i,k}$ is the mean depth of pixel i in cluster k. Further, $G_{i,j;\sigma}$ is a spatial Gaussian kernel with standard deviation σ. Intuitively, we count the number of cluster assignments modulated by a weight that depends on proximity both in image space and in depth, where the parameter α steers the depth weight falloff. The final color c_i is obtained via

$$\bar{c}_i = \frac{\sum_k W_{i,k}^{\gamma} c_{i,k}}{\sum_k W_{i,k}^{\gamma}}$$

The parameters σ, α, and γ are determined using a parameter sweep (Sec. 3.7). This filtering approach eliminates many of the incorrect PVM artifacts observed in PVM-H, and in some cases improves over Deep Blending (Hedman et al. [2018]).

3.6 Hybrid Rendering

Our hybrid rendering algorithm first renders the novel view using textured mesh (TM), since this is required by all methods (ULR and PVM). We then select which algorithm should be used for each pixel, building on the analysis presented in Sec. 3.3.3.

3.6.1 Hybrid Rendering Algorithm

Our hybrid approach follows the flow chart shown in Fig. 3.6. We first decide whether a pixel is outside the PVM grid bounds; in this case it is rendered with ULR. We then decide whether a pixel can be rendered with the global mesh based on *geometric uncertainty*. Regions with high geometric uncertainty should be rendered with PVM, and others with the global mesh. For pixels that can be rendered with the global mesh, ULR should be preferred in view-dependent regions. To find these, we use a measure of underlying surface roughness using *photometric uncertainty* for a given pixel in the novel view. Uncertain pixels are assumed to be view-dependent and thus are rendered with ULR while pixels with low uncertainty are assumed to be diffuse and are rendered with TM. This assumption comes from the observation that surfaces which exhibit glossiness are highly uncertain while diffuse surfaces exhibit low uncertainty. We next provide details on how we compute geometric and photometric uncertainty.

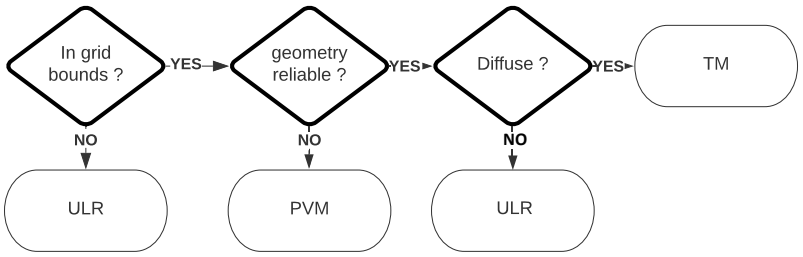


Figure 3.6: Hybrid Algorithm Selection Flow Chart

Geometric Uncertainty

To measure the geometric uncertainty at a pixel we compare PVM depth with the global mesh. Specifically, any fragment which exceeds the absolute depth difference by $3cm$ is rendered with PVM. A threshold of $3cm$ works best empirically for all our datasets. Depth differences lower than this threshold imply the global depth is close enough to per-view depth and the region can be reliably rendered using the global mesh.

Since PVM tiles have multiple depths, we use the clustering step described above (Sec. 3.5.2) to determine which depth to use for this comparison. If d_{fg} is the depth of the foreground cluster and d_{bg} the background, we estimate depth using a voting strategy

$$d = (\#(d_{bg}) >= \tau_{depth} * \#(d_{fg}))\ ?\ d_{bg} : d_{fg} \tag{3.2}$$

where $\#$ is the size of each cluster, and τ_{depth} the depth threshold.

Photometric Uncertainty

For the remaining pixels that can be rendered with the global mesh, our main criterion is whether they correspond to a diffuse region or not. We have already computed this information as part of the image harmonization method (Sec. 3.4.2). Specifically, we have a binary mask of view-dependent regions for each input image. We use these masks to obtain a texture map indicating view-dependent regions over the global mesh by computing the weighted average of the mask values accumulated per-vertex, which we call *view-dependent texture*. The texture map is computed in a pre-process after image harmonization. We perform per-pixel ULR for pixels determined as highly view-dependent (i.e., variance $> \tau_{var}$) and use TM for the rest.

Parameters τ_{var} and τ_{depth} are determined using a parameter sweep on a synthetic scene (Sec. 3.7).

3.6.2 Rendering

The rendering loop proceeds as follows:

1. Rasterize global TM, select voxels for tile selection, mask background pixels if out of grid bounds.
2. Rasterize PVMs: rank tiles and create up to 12 layers per pixel.
3. Clustering and Blending Shader: perform clustering step, create mask deciding PVM/TM or ULR.
4. **If** ULR **then**

   ULR blend shader: apply per-pixel ULR
5. **Else if** PVM **then**

   Spatial Filter Shader: output filtered color
6. Blur the mask between algorithms and composite

We blur the mask with a kernel of size σ_{blur}, also determined by our parameter sweep (Sec. 3.7). The mask is visualized in Fig. 3.7, last row. We implement our hybrid rendering algorithm in our C++ framework using OpenGL/GLSL (Bonopera et al. [2020]). The source code along with all pre-processing utilities can be found here: `https://gitlab.inria.fr/sibr/projects/hybrid_ibr`

3.7 Results and Evaluation

We provide comparisons of our rendering method with baseline algorithms as well as current state-of-the-art neural rendering methods, using published codes (see Appendix A for details). We also perform baseline comparison and ablation studies to show the gain in quality achieved with harmonization as well as PVM-H rendering. We provide a supplemental video, and a supplemental webpage on the project page provided above in Sec. 3.6.2 which provides all comparisons available.

Parameter evaluation

To select values for parameters for the filtering and hybrid rendering steps we selected 22 novel views in a synthetic scene. We computed the ground truth using path tracing and performed a sweep of acceptable values selecting the average of the values that gave the lowest error. The values are $\sigma = 3$, $\alpha = 7$, $\gamma = 7$ for the filter (Sec. 3.5.2), $\tau_{depth} = 0.83$, $\tau_{var} = 0.82$ (Sec. 3.6.1), and finally $\sigma_{blur} = 4$ (Sec. 3.6.2).

Datasets

We present our results on 6 datasets from Hedman et al. [2016]: *Dr Johnson, Hugo, Library, Playroom, Creepy Attic* and *Ponche*, 1 dataset from Riegler and Koltun [2020]; Knapitsch et al. [2017]: *Train*, 1 new indoor dataset that we captured: *Salon*, and 1 synthetic dataset: *Synthetic Attic*. We present 6 indoor and 3 outdoor scenes, showing the versatility of our method.

3.7.1 Rendering Results & Comparisons

Fig. 3.7 shows the comparison between our hybrid rendering and the following algorithms (see supplemental webpage and video):

- Textured Mesh(TM): Global mesh textured with Reality [2018]; Jancosek and Pajdla [2011].
- Per-pixel ULR as described in Sec. 3.3.3.
- Inside Out: SfM+MVS variant of Hedman et al. [2016] used as a baseline in Hedman et al. [2018].
- Deep Blending: Method of Hedman et al. [2018] with learned blend weights.
- Free View Synthesis (FVS): learning-based method of Riegler and Koltun [2020].

We also compare with Neural Radiance Fields (NeRF), a neural scene representation method of Mildenhall et al. [2020] for the Salon scene. Directly using the full set of images did not give good results, so we present a best-effort attempt by selecting a subset of images that gave the best results.

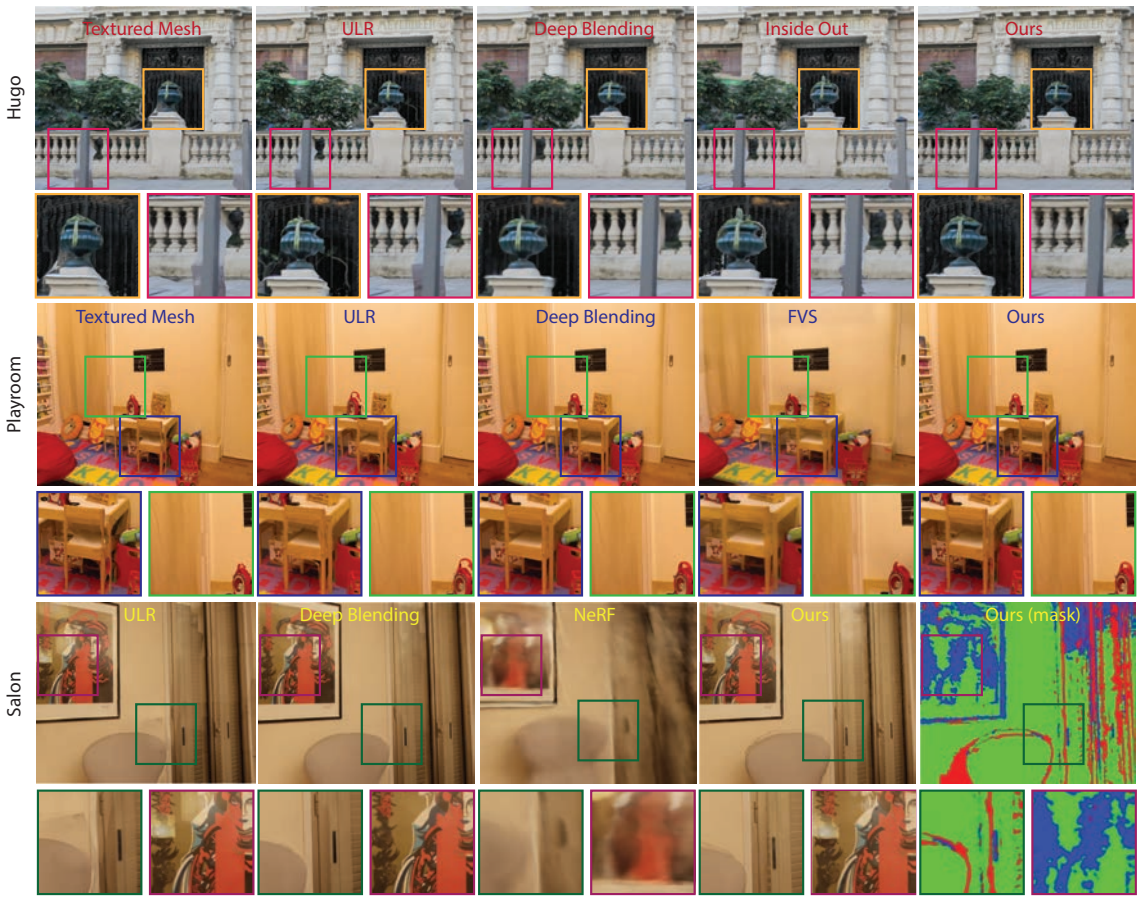


Figure 3.7: Comparison of novel view rendering results in Hugo, Playroom, and Salon. We show comparison against different classical IBR (Textured Mesh, Unstructured Lumigraph (Buehler et al. [2001]), Inside Out (Hedman et al. [2016])) and neural rendering (Deep Blending (Hedman et al. [2018]), NeRF (Mildenhall et al. [2020]), Free-view Synthesis (Riegler and Koltun [2020])) approaches. Our method achieves better quality by removing many common IBR artifacts such as color seams (bottom row, dark green inset), view-dependent effects preservation (bottom row, magenta inset), missing geometry (middle row), over-reconstructed geometry (top row, pink inset), texture sharpness and ghosting artifacts (top row, orange inset).

Our method corrects erroneous geometry compared to TM while maintaining sharp details in diffuse regions. Compared to ULR we often reduce ghosting and blurring artifacts as well as prominent harmonization artifacts due to blending. Our two-level PVM with filtering improves visual quality compared to Inside Out. Compared to neural methods of Deep Blending and FVS, our method provides more stable and often sharper results, removing most of the visible color artifacts encountered due to neural rendering.

Figure 3.8: Comparing our solution with PVM-H in Library (left) and Dr Johnson (right).

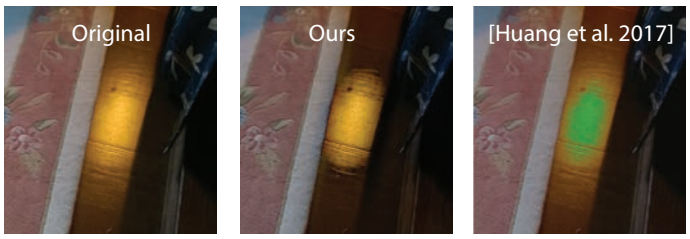


Figure 3.9: Evaluation of Image Harmonization on Library.

Note however that FVS was trained on the very different Tanks and Temples dataset of Knapitsch et al. [2017].

In terms of pure rendering quality, Deep Blending is arguably better in many cases, but at more than 2.5 times slower frame rate (see below). Our approach provides good balance in the quality/speed tradeoff, and often is better at maintaining high frequencies.

As noted in Sec. 3.3.2, PVMs can create visible artifacts in extreme cases which can be hard to solve even using neural networks. Our two-level PVMs with filtering provide significant improvement in such hard cases. Fig. 3.8 compares our PVM blending algorithm with PVM-H method of Hedman et al. [2018]. Our algorithm is able to get rid of most such cases and provides much cleaner and sharper edges.

Image Harmonization: Baseline comparison

As discussed in Sec. 3.2, many methods perform color correction to avoid visible seams in the textures. We implement the optimization-based solution of Huang et al. [2017] for color correction as a baseline comparison to our method. The method recovers a single parametric curve per-channel per-input image and scales the intensity of every pixel in the image based on its recovered curve. While this helps achieve harmonized color in most diffuse regions, the regions which exhibit specular highlights result in color artifacts, see Fig. 3.9.

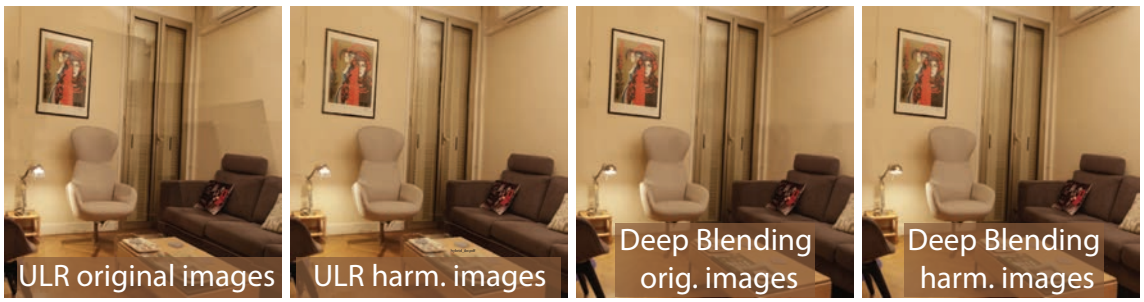


Figure 3.10: Ablation results of Image Harmonization on Salon.

Harmonization for ULR and Deep Blending

Harmonization can be useful for other algorithms. We show results using our harmonized and original images for ULR and Deep Blending in Fig. 3.10. Basic Deep Blending tries to harmonize the images by learning the blend weights such that they result in fewer seams but fails to completely remove it, while our image harmonization removes the seams completely.

3.7.2 Performance

We ran tests on a desktop machine, with Intel Xeon Gold 5218 2.30GHz Processor and Quadro RTX 5000 GPU and a laptop with an Intel Core i9-8950HK 2.90GHz Processor and NVIDIA GeForce RTX 2080 Max-Q GPU. All timings are on the desktop unless stated otherwise.

Table 3.2: Average cost of each step for our rendering algorithm for Ponche & Library.

Steps	Ponche	Library	Steps	Ponche	Library
1. Voxel lookup	3.06 ms	3.11 ms	5. Spatial Filtering	1.99 ms	1.29 ms
2. Tile Sort	9.83 ms	4.29 ms	6. ULR Blend	0.94 ms	1.35 ms
3. Per-View Depth Pass	5.30 ms	4.00 ms	7. Masking	2.01 ms	1.34 ms
4. Cluster & Blend	5.42 ms	5.07 ms	**Total**	**28.55 ms**	**20.45 ms**

Runtime Statistics:

Table 3.2 provides a runtime breakdown of different steps of our algorithm. We provide additional runtime comparisons on the Desktop and Laptop machines and pre-processing time statistics in Appendix A.3. Pre-processing for the harmonization takes between

15 min to 2-3 hours, depending on the number of images in the dataset and their resolution; the MRF step is the most expensive.

3.7.3 Quantitative Evaluation

For completeness, we present quantitative evaluations on a synthetic and a real scene. As discussed below, current quantitative metrics are not very successful at identifying the kind of visual artifacts created by free-viewpoint IBR algorithms.

The Synthetic Attic dataset is shown in Fig. 3.11. We show path-traced ground truth and comparisons with our method as well as previous approaches. We also show the algorithm selection mask for the given input view which indicates the algorithms used for each pixel. Notice how our hybrid algorithm is able to identify the occlusion edges as the regions where most geometric errors occur.

We performed a quantitative analysis on the real-world dataset Ponche (Hedman et al. [2016]), by holding out 10% of the input views for rendering. For ULR, all test images were left out. For Inside Out, Deep Blending, and our method the corresponding held-out per-view meshes are not used. However, it should be noted that during the global mesh reconstruction and texturing the held-out images were used.

We also provide a quantitative comparison with path-traced ground truth images on the Synthetic Attic scene in Table 3.3. All test images were unseen during the entire pre-processing (including camera calibration and reconstruction) for the synthetic dataset.

Table 3.3 reports quantitative error of our method compared to previous IBR algorithms. We show structural dissimilarity (DSSIM) (Loza et al. [2006]), E-LPIPS perceptual metric (Kettunen et al. [2019]) and Peak Signal-to-Noise Ratio (PSNR).

The quantitative results indicate that, for the real scene Ponche, we have achieved our goal of having equivalent, if not – slightly – better quality than previous methods, but with faster compute time. While we perform best with respect to the E-LPIPS and PSNR metrics in the Ponche real-world dataset, Deep Blending (Hedman et al. [2018]) performs best across the 3 metrics in the synthetic scene. According to DSSIM, ULR performs better than both our and Deep Blending in the real-world dataset; This is in contrast to what we perceive from the video sequences (please see supplemental) as ULR tends to have numerous color seams, ghosting, and blurring artifacts compared to all other methods. In addition, the differences between algorithms are generally very small, again

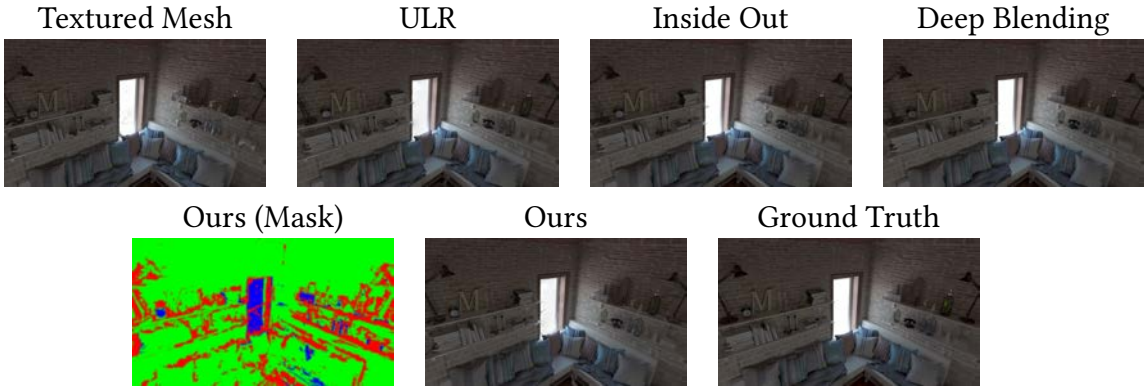


Figure 3.11: Comparisons with path-traced ground truth rendering for each method on Synthetic Attic dataset. Top Row - Baselines: Textured Mesh (Reality [2018]), Unstructured Lumigraph (ULR) (Buehler et al. [2001]); Previous methods: Inside Out (Hedman et al. [2016]), Deep Blending (Hedman et al. [2018]); Bottom row - Ours per-pixel algorithm mask (Red: Per-view Mesh (PVM); Blue: Unstructured Lumigraph (ULR); Green: Textured Mesh (TM)); ours and the path-traced ground truth.

Table 3.3: Quantitative comparison of different algorithms over images rendered in held-out fashion for real dataset Ponche while using ground truth path-traced images of novel views for synthetic dataset Synthetic Attic.

IBR Algorithms	Ponche			Synthetic Attic		
	E-LPIPS ↓	DSSIM ↓	PSNR ↑	E-LPIPS ↓	DSSIM ↓	PSNR ↑
TM	0.0296	0.150	21.26	0.0225	0.160	25.29
ULR	0.0243	**0.123**	21.59	0.0234	0.147	26.38
InsideOut	0.0290	0.129	21.78	0.0221	0.143	27.90
Deep Blending	0.0279	0.132	22.04	**0.0203**	**0.137**	**28.24**
Ours	**0.0239**	0.138	**22.06**	0.0218	0.154	26.85

in contrast to visual evidence. Given these inconsistencies and unreliability, quantitative comparisons are of questionable utility in this context, underlining the need for further research on perceptual metrics specific to different types of IBR artifacts – both at global and local scales.

3.8 Limitations and Conclusions

We analyzed common IBR artifacts and used the insights obtained to propose a hybrid rendering algorithm that runs at interactive rates, with a good balance between speed and quality. Specifically, we have shown how to *(i)* perform effective image harmonization

while preserving view-dependent effects, *(ii)* utilize and fuse per-view information to correct for geometric reconstruction errors, and *(iii)* combine the strengths of three different algorithms in a novel hybrid algorithm.

Limitations: While we show that we indeed can leverage the strengths of the individual algorithms, residual artifacts remain due to their weaknesses. Our hybrid approach often chooses TM, that can result in deformed lines (see Playroom sequence, 0:16) or over-reconstructed geometry on object edges (see Salon sequence, 0:12 and supplemental video 0:05). ULR can be prone to ghosting artifacts when the silhouette of objects is not reconstructed accurately; our filtered PVMs occasionally exhibit (semi-)transparent foreground regions, whenever our cluster assignment assumptions are violated. Further, we do not explicitly enforce temporal consistency in our PVM module. Extending our depth filtering approach to the temporal domain could be a direction for future work, but would require a more elaborate treatment of fuzzy depth.

Future Work: While we focused on existing, interactive rendering algorithms, we believe that our hybrid design in principle allows the exploration of other combinations of existing or future algorithms. When neural rendering becomes sufficiently fast, it could provide missing high-quality details for specific image regions our framework is able to identify through simple yet effective heuristics.

We have presented an IBR algorithm to capture and render a real scene with a good speed/quality tradeoff. In the process we used two different geometric models obtained from the photographs: MVS *global* and *per-view* meshes. While a scene can be rendered with the reconstructed mesh and photographs, the scene is not editable. The rough appearance estimate we obtained using photometric uncertainty gives an idea of view-dependency of underlying surface but is not sufficient to re-render the scene with modifications. To edit the scene, we require to factor it into mesh, material appearance and/or lighting and re-render with one of the three components modified. As we saw, we have automatic solutions to reconstruct the mesh from images, but the global mesh is reconstructed with a diffuse surface assumption. In the next chapter we explicitly recover the surface material properties using the MVS mesh and the photographs to allow editing of real captured scenes by introducing a deep-learning based approach.

Chapter 4

Material estimation from indoor captures

Rendering a real scene with edited contents is a long sought-after task in computer graphics. In the previous chapter we achieved a good quality/speed tradeoff by rendering a scene with a rough estimate of the underlying surface appearance. But such a rough estimate is not enough to edit a scene. The way researchers approach this task is by creating a model of the scene using photographs and/or user inputs. Photogrammetry is the technology of creating renderable scene assets from captured photographs. The movie and video game industries have adopted photogrammetry as a way to create digital 3D assets from multiple photographs of a real-world scene. But photogrammetry algorithms typically output an RGB texture atlas of the scene that only serves as visual guidance for skilled artists to create material maps suitable for physically-based rendering.

In this chapter we present a learning-based approach that automatically produces digital assets ready for physically-based rendering, by estimating *approximate material maps* from multi-view captures of indoor scenes that are used with retopologized geometry. Due to the success of learning-based approaches for material estimation on surface patches, we base our approach on a material estimation Convolutional Neural Network (CNN) that we execute on each input image. We leverage the view-dependent visual cues provided by the multiple observations of the scene by gathering, for each pixel of a given image, the color of the corresponding point in other images. This image-space CNN provides us with an ensemble of predictions, which we merge in texture space as the last step of our approach. Our results demonstrate that the recovered assets can be directly used for physically-based rendering and editing of real indoor scenes from any viewpoint and novel lighting. Our method generates approximate material maps in a fraction of time compared to the closest previous solutions thus significantly reducing the time required to generate material assets compatible with traditional path tracers.

4.1 Introduction

While physically-based rendering is now a mature technology (Fascione et al. [2017]), creating the digital assets to be rendered remains a major bottleneck in the creative industry. Photogrammetry has gained popularity to create digital assets from real-world scenes. A popular workflow consists in first capturing multiple photographs of the scene, then using multi-view stereo algorithms to compute an approximate 3D model from these photographs. The approximate geometry is then manually edited to create models compatible with traditional rendering pipelines – a task known as *retopology* (Lachambre et al. [2017]).

Unfortunately, existing photogrammetry solutions typically output a simple RGB texture of the scene with baked-in lighting, which only serves as a crude initialization for artists who need to create rich materials maps used by downstream physically-based renderers. Creating these maps involves significant manual work, including removing shading, shadows and highlights to form the diffuse albedo, and guessing specular strength and roughness parameters over different surfaces. We propose a learning-based approach that addresses this difficult task by augmenting the photogrammetry workflow by *automatically* estimating approximate material maps from multiple photographs of an indoor scene. Our goal is to provide assets that directly allow plausible renderings of the captured scene. Specifically, the output of our method are approximate Spatially-Varying Bidirectional Reflectance Distribution Function (SVBRDF) *texture atlases* which, combined with retopologized geometry, forms a digital asset ready for physically-based rendering of indoor scenes. We call these *material maps* from now on.

Learning-based methods for material estimation have focused on pictures of flat surface patches (Deschaintre et al. [2018]; Guo et al. [2021]) or on *single images* of isolated objects (Li et al. [2018b]) and scenes (Li et al. [2020]), for which the prediction can be efficiently performed in image-space using convolutional neural networks (CNNs). In contrast, approaches based on inverse rendering compute accurate material parameters in object or texture space (Yu et al. [1999]; Nimier-David et al. [2021]) to benefit from observations from multiple viewpoints. But the underlying optimization is expensive and needs to be recomputed for every new scene. We present the first method that combines ideas from these two streams of research. On the one hand, we leverage the strength of image-space CNNs to predict approximate material parameters for each photograph of the

scene. On the other hand, we exploit multi-view information by gathering, for each pixel in a photograph, observations of the same scene point in other photographs. Furthermore, we aggregate the predictions given by each photograph into a common texture space to form the final texture atlas. Our method thus offers the speed of learning-based material estimation previously applied for single images, for the much harder *scene-scale* material estimation problem.

Our method addresses several difficulties raised by the long-standing challenge of scene-scale material estimation. First, in contrast to single-image methods (Deschaintre et al. [2018]; Li et al. [2018b, 2020]), our multi-view setting receives a varying number of observations per pixel to be processed by the CNN. We overcome this difficulty by computing a fixed number of color statistics, which forms the initial feature maps that we feed to the CNN. Second, photographs of indoor scenes exhibit complex interactions between geometry, lighting and materials via indirect illumination. Prior work on inverse rendering model these interactions using approximate global illumination (GI) to jointly recover shape, materials and light (Li et al. [2020]). In contrast, we consider a scenario where geometry is reconstructed with photogrammetry and retopology, such that we only need to recover material appearance. Rather than approaching this as an inverse rendering problem where global illumination must be estimated accurately to match the input observations, we train an illumination agnostic network to produce materials maps of similar appearance to the inputs. To compare the predictions to the ground truth materials, we use a rendering loss which only operates via local camera-space relighting, avoiding the underconstrained estimation and expensive computation of GI altogether. The third challenge is building a synthetic training dataset suitable for scene-scale approximate material estimation; we created a dataset from professionally modeled scenes, and provide a framework that allows the generation of new datasets for this task.

In summary, our contributions are:

- A deep neural network architecture for material estimation that exploits scene-scale multi-view input.

- A proof-of-concept solution allowing fast, scene-scale material estimation to produce digital assets suitable for physically based rendering and editing of real indoor scenes, that integrates seamlessly into the current photogrammetry workflow.

- A scene-scale synthetic dataset with ground truth SVBRDF maps and the tools to generate it, used to train our multi-view material estimation network.

We evaluate and illustrate our method on synthetic scenes that allow quantitative analysis of our algorithmic choices, and show first results on captured real scenes. We demonstrate that our automatically estimated material map atlases – albeit approximate – are of sufficient quality to allow physically-based rendering of the captured scene with novel lighting conditions and scene editing (see Fig. 1.4(c), 4.8). We provide the source code to our system, including all the tools required to generate the training dataset from commercially-available models on our project webpage: `https://repo-sam.inria.fr/fungraph/deep_multiview_scene_materials/`.

4.2 Related Work

As we discussed in Sec. 2.3.3 material capture makes up a significant part of computer graphics research. To keep our discussion relevant, here we summarize the most recent advances in lightweight material capture using optimization-based and learning-based methods highlighting the differences with our method.

4.2.1 Optimization-based material capture

With the advances in differentiable inverse rendering, many methods such as Azinovic et al. [2019]; Nimier-David et al. [2021]; Haefner et al. [2021] recover scene-scale material and lighting by solving complex global illumination effects. We see our approach as complementary to such optimization-based algorithms that work on scene-scale. On the one hand optimization-based algorithms are capable of recovering more precise information by minimizing the difference between the input images and the images re-rendered from the estimated materials. On the other hand, such a minimization typically takes 10-12 hours to converge due to complex global illumination computations and is highly sensitive to initialization. Our approach could speed-up these optimization methods by providing an initialization that is much closer to the end result compared to the random material maps that are typically used. Other methods such as Nam et al. [2018]; Goel et al. [2020]; Luan et al. [2021]; Bi et al. [2020] also optimize for geometry but have been restricted to isolated objects only. In contrast, we take multiple unconstrained sparse viewpoints of the scene resulting in a variable number of observations for different

scene regions and complex visibility issues due to inexact geometry.

4.2.2 Learning-based material capture

Taking advantages of the practical solutions which machine learning provides, several methods have been developed to estimate per-pixel material properties of flat surface patches (Deschaintre et al. [2018]; Gao et al. [2019]; Guo et al. [2021]; Henzler et al. [2021]), isolated objects (Li et al. [2018b]; Boss et al. [2020]) and even of indoor scenes (Li et al. [2020]) from a single image. Ours is the first method to take wide baseline, scene-scale multi-view input under unknown indoor lighting for material estimation. Most related to our goal is the concurrent work by Li et al. [2022] for indoor scene material prediction but our work is complementary, as we explore material prediction in indoor scenes under a multi-view capture scenario as opposed to single-view prediction.

Our key insight is that the multiple images that are typically captured for photogrammetry offer complementary observations of material appearance. As discussed earlier, multi-view information needs proper aggregation before feeding to a CNN and assembly to form a valid texture atlas for later use in rendering engines. Methods which focus on planar patches (Deschaintre et al. [2019]; Guo et al. [2020]; Asselin et al. [2020]; Ye et al. [2021]) do not suffer from this problem since it's assumed that each pixel of the surface is visible in all views. This is not the case when dealing with complex scenes where parts are frequently occluded or out of the field of view of many of the input photographs. We propose a solution based on image re-projection and pixel statistics to process multi-view inputs with a standard CNN architecture, and to merge multiple predictions into a single texture atlas.

Neural representations recently emerged as an effective solution to relight 3D content captured from multiple photographs (Srinivasan et al. [2021]; Zhang et al. [2021b,a]; Boss et al. [2021a,b]). However, these novel representations are not compatible with the well-established photogrammetry workflow, where artists seek to create triangular meshes and texture atlases compatible with downstream industry-standard rendering engines. While Philip et al. [2021] also feeds color statistics as one of the many multi-view information to a neural renderer for novel-view synthesis with relighting, we predict explicit material parameters in the form of material texture maps and we assemble the predictions given by multiple views in a common texture space which is readily available to the user for further editing as desired. This post processing flexibility is missing from

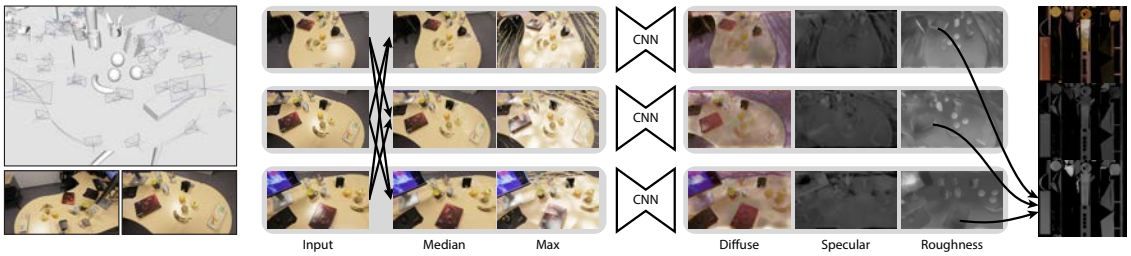


Figure 4.1: Overview of our method. We take a multi-view dataset, calibrated cameras and a retopologized mesh as input (a). We re-project information from multiple views into each input view to compute color statistics for each observed point (b). We feed this multi-view information into a convolutional neural network to predict material maps for each view (c). We finally merge this ensemble of inferred maps into texture space (d) to form a scene-scale material texture atlas.

prior works. Munkberg et al. [2021] recovers a triangle mesh, an SVBRDF texture and an environment map but their approach has only been demonstrated on isolated objects.

Training methods like ours require large amounts of photorealistic images with ground truth material map labels. Such a dataset is infeasible to capture so we rendered visually realistic synthetic scenes with variations in lighting, materials, geometry and viewpoints. While several datasets of indoor scenes have been described, many only provide images rendered from pre-defined viewpoints and do not allow the generation of new images, as is the case for *OpenRooms* (Li et al. [2021]). Other datasets do not include the labels we are interested in, such as *Hypersim* (Roberts et al. [2021]) that provides diffuse albedo maps and a non-diffuse residual term, which is not directly compatible with existing BRDF models suitable for physically-based renderers. We built a dataset tailored to multi-view material estimation in indoor scenes, by developing an asset generation system that assembles objects from synthetic scenes modeled in Autodesk 3DS Max and then rendering with Mitsuba (Jakob [2010]). We hope our dataset generation tools will help foster research on scene-scale material estimation and other scene-scale learning-based tasks.

4.3 Overview

Our method takes as input multiple casually-captured unstructured wide-baseline images of a scene from different viewpoints using a DSLR camera. This results in multi-

view observations, but in some cases capture is incomplete: In particular, lighting and some parts of the scene might be unobserved. Similar to commercial photogrammetry pipelines (Lachambre et al. [2017]), first we obtain camera calibration and a rough multi-view stereo mesh using RealityCapture (Reality [2018]), followed by re-topology where an artist creates a clean version of the mesh, suitable for rendering (see Fig. 4.1(a) and Fig. 4.5). This reliable 3D geometry is used as input to our method. Given the multiple input images and the corresponding geometry, our goal is to produce an atlas of material parameters, i.e., spatially-varying diffuse albedo D, specular albedo S, and roughness R for a Cook-Torrance BRDF model (Cook and Torrance [1982]). We do not estimate normal maps since we focus on indoor scenes composed of large surfaces seen at a distance, for which our retopologized geometry provides sufficiently accurate normals (see Fig. 4.1(a) top and Fig. 4.5).

We achieve this material estimation task in two main steps, illustrated in Fig. 4.1 (b - d). The first step relies on an image-space CNN to predict material maps for each input image separately. For each view, we use the 3D geometry to reproject image colors from other views and deduce color statistics (minimum, median and maximum color) that summarize the view-dependent appearance of each pixel. We complement these statistics with geometry buffers (surface normals and depth). In practice, we found it beneficial to split the prediction task into two tracks, one responsible for the prediction of diffuse albedo and one responsible for the prediction of specular albedo and roughness.

The second step of our method aggregates the per-view predictions into a common texture space to form the final atlas. We use simple median filtering to select a consensus from the ensemble of predictions given by all views where a surface point appears. Mapping this atlas onto the retopologized mesh gives a complete asset that is compatible with traditional physically-based rendering (see Fig. 1.4(c), 4.8), including full editing capabilities such as changing the lighting and inserting new objects.

4.4 Multi-View Aware Deep Material Estimation

Our problem is estimating *scene-scale* material properties from a *multi-view dataset* under *unknown lighting*, as opposed to the several successful deep learning methods for estimating SVBRDF maps: These start from one or a few images (Li et al. [2017, 2018a]; Deschaintre et al. [2018]; Guo et al. [2021]; Zhou and Kalantari [2021]; Deschaintre et al.

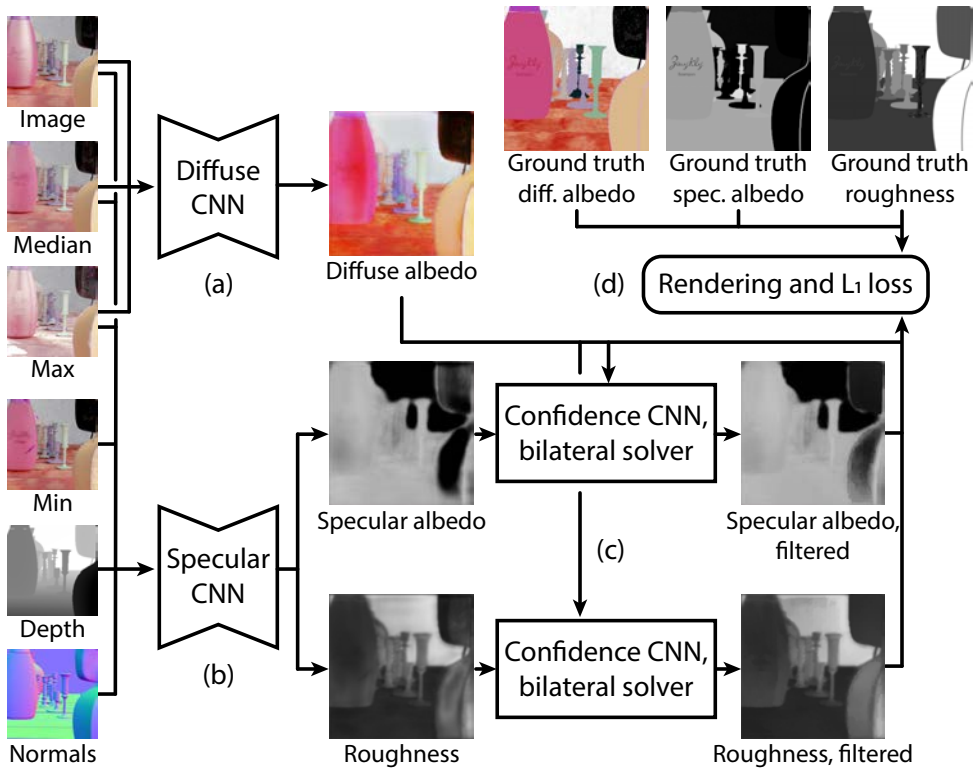


Figure 4.2: Overview of our deep learning architecture and training procedure. We split the material estimation task into two tracks, one for the diffuse albedo and the other one for specular albedo and roughness (a, b). In addition, we filter the specular albedo and roughness with a bilateral solver (c). Finally, we compare the predicted maps with ground truth maps using a $\mathcal{L}_1$ loss as well as with a rendering loss that assesses the appearance of the materials under several lighting and viewing conditions (d).

[2019]; Asselin et al. [2020]), typically for small planar patches of materials lit by a flash. We tackle the scene-scale material estimation problem by first processing each view with a CNN similar to the one used by Deschaintre et al. [2018, 2019]; We explain how we adapted this architecture to our use-case in Sec. 4.4.2.

The much harder scene-scale problem precludes the use of co-located flash lighting; Since we cannot benefit from the rich visual cues given by this mode of capture, our originality is to instead leverage visual cues provided by multi-view observations. Fig. 4.2 illustrates the main components of our architecture and its training procedure.

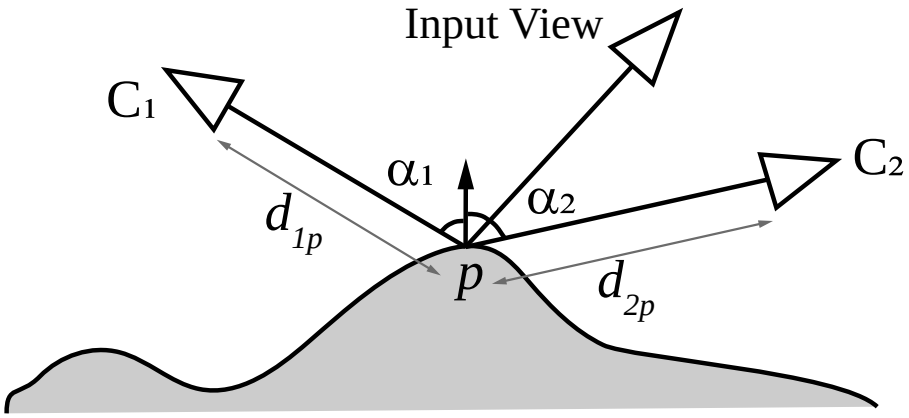


Figure 4.3: Camera selection for each surface point p. A cost function is used to select the 12 best views that consists of a visibility term and a distance term which favors normal-aligned views and cameras close to the surface.

4.4.1 Network Inputs

While we run our CNN on each input view separately, we feed it with multi-view information obtained by re-projecting a set of neighboring views to the current view. For each pixel, we select at most 12 views where the corresponding point is visible, and for which the view direction is most closely aligned to the surface normal, as those views are less prone to grazing angle observations. We also add a distance term to favor cameras nearer to the surface.

We rank and pick the top-12 views by minimizing the cost composed of a view term and a distance term as shown in Fig. 4.3, specifically for an observation at p:

$$\text{cost}_i \;=\; cos\ \alpha_i + \frac{d_{ip}}{max(d_{jp})} \qquad \forall j \in \{1, N\} \tag{4.1}$$

where d_{ip} is the distance of point p to camera i, N is the total number of cameras, and α_i is the angle between the normal at surface point p and view direction of camera i.

Since each pixel might receive a different number of observations as not all surface may be visible in 12 or more views (this is frequently observed in the case of background pixels), we summarize this multi-view information as a fixed set of images corresponding to simple color statistics, i.e. median, maximum and minimum.

The maps that form a material map atlas have different characteristics, and can be inferred from different visual cues. On the one hand, the diffuse albedo map needs to maintain

sharp texture features while being free of view-dependent highlights, shadows and indirect light. On the other hand, the specular albedo and roughness values are often nearly constant over parts of objects made of the same material, and are conveyed by highlight information from different views. These observations motivate us to predict the diffuse and specular parameters via two different tracks, and to feed each track with different visual information.

For the diffuse albedo, we complement the input image with the median image, as the median rejects highlights that only appear in a few images. We also include the maximum image to help the network locate shiny areas where highlights might need to be suppressed even if present in many images.

In addition to the images used by the diffuse track, we feed the specular track with the minimum image as it further helps locate parts where the color changes significantly across views. We also provide the specular track with normal and depth maps, which delineate different objects that often have different specular values. Our experiments revealed that providing all these extra images to the diffuse track degrades its prediction, as the network struggles to select the relevant image structures among too many visual channels. In addition, reprojection errors due to the approximations in retopologized geometry can be problematic (see Sec. 4.7.1).

Finally, we favor smooth specular maps by post-processing their predictions with a differentiable bilateral solver (Barron and Poole [2016]; Li et al. [2020]) guided by the predicted diffuse albedo. This edge-aware smoothing attenuates discontinuities due to reprojection misalignments. Unlike Li et al. [2020] we do not run our diffuse albedo maps through the solver as we observed this leads to over-smoothing of diffuse albedo maps resulting in loss of texture details.

4.4.2 Network Architecture

Our network architecture is based on the ones by Deschaintre et al. [2018, 2019] which were also designed for material estimation but works on a single flat patch of size $256 \times 256 \times 3$. Their architectures follow the widely popular U-Net encoder-decoder (Ronneberger et al. [2015]), to which a fully-connected track responsible for processing and transmitting global information was added. We maintain the same U-Net architecture augmented with the global track but we half the number of feature layers from 8 to 4 to

make the networks lighter with the feature counts in the encoder downscaling layers of 64, 128, 256, and 512. We did not observe any significant degradation in the maps due to this reduction in network capacity. We follow the same downsampling and upsampling process as described in Deschaintre et al. [2018] with the feature counts used in reverse order for the decoder. Instance normalization is used for stability and we also regularize by applying dropout at 50% probability on the last three layers of the decoder.

The main difference lies in the input/output; Specifically our network architecture differs in two ways. First, we separate out the network into two tracks one each for diffuse and specular maps. The two tracks serve different purposes with the diffuse track predicting only diffuse albedo maps, while the specular track outputs the specular albedo and roughness maps. The two tracks are two separate networks. While training, we can either train one of the networks without sharing any information between them or both networks by jointly computing the loss on output of both networks. We use individual training to train the networks and joint training to fine-tune the networks post-training. Please see Sec. 4.6 for training details. Second, since we feed the network with multi-view information in the form of image statistics, we have more input channels compared to 3 channels for the previous networks. Concretely, since we feed the median, maximum and input image to the diffuse track which thus has 9 input channels, while for the specular track, we also provide the minimum image, depth and normal buffers resulting in 18 input channels.

For the network used to predict confidence channels to guide the bilateral solver, we follow the same CNN architecture and hyper-parameters for the solver as used in previous works (Li et al. [2020]; Barron and Poole [2016]). We include detailed breakdown of the network architecture and the parameters used in Appendix B.1.

4.4.3 Loss Function

Many inverse rendering methods supervise their predictions of geometry, material and lighting by comparing re-rendered images to input images using a differentiable *rendering loss* (Li et al. [2018b]; Boss et al. [2020]). At scene-scale, such a rendering loss needs to model global illumination effects present in the input (Li et al. [2020]). We depart from this family of methods by focusing on a scenario where geometry is given, such that our task boils down to predicting material maps only. In this context, we can supervise our method by comparing our prediction to ground truth SVBRDF maps rather than by

attempting to reproduce the input. A local lighting model is sufficient for this purpose, as was originally proposed by Deschaintre et al. [2018] in the context of planar surface patches. While Deschaintre et al. use point and directional lights, we improve on their approach by incorporating distant area lights modelled as spherical Gaussians.

Concretely, we use a simple differentiable renderer that takes as input the material maps along with the normals of the geometry. Our goal is to compare the local appearance of our predicted material maps to the local appearance of the ground truth SVBRDFs. To do so, we render the prediction and the ground truth under several viewing and lighting conditions and compare the resulting images under the $\mathcal{L}_1$ norm and E-LPIPS perceptual metric $\mathcal{L}_E$ (Kettunen et al. [2019]) after applying a log transform ($I' = log(0.1 + I)$) to compress the dynamic range of the renderings. Following Deschaintre et al., we generate random viewing conditions by sampling view vectors over the hemisphere centered around the original view direction from which the input image was rendered. We then generate lighting conditions likely to produce highlights by positioning a point light in the mirror direction of the view vector. Finally, we also create extended light sources by generating a mixture of 5 Gaussian lights with random width, color and direction distributed over the hemisphere.

We implement the shading of a point under distant area light sources by using the spherical warp introduced by Wang et al. [2009]. The light, as well as the BRDF, are approximated as two Spherical Gaussians, for which a fast closed-form convolution exists. Using this approximation allows us to include extended light sources in the rendering loss without losing computational efficiency for training. Our final rendering loss averages the image differences obtained with three point-wise lighting conditions and with three extended lighting conditions.

In addition to the rendering loss, we also use $\mathcal{L}_1$ and $\mathcal{L}_E$ to compare the individual predicted maps to their respective ground-truth. Denoting I a rendered image, D the diffuse albedo, R the roughness, and S the specular albedo, the total loss we use is thus:

$$\begin{aligned}\mathcal{L} = &[\mathcal{L}_E(I) + \mathcal{L}_1(I)] \\ &+ \mathcal{L}_E(D) + \mathcal{L}_E(R) + \mathcal{L}_E(S) \\ &+ \lambda(\mathcal{L}_1(D) + \mathcal{L}_1(R) + \mathcal{L}_1(S))\end{aligned} \tag{4.2}$$

where λ is 0.1.

4.4.4 Merged Renderable Scene Assets

The second step of our method merges the material maps predicted over each input view to form a single, object-space material map texture atlas suitable for rendering. We leverage the retopologized 3D mesh to identify which texel corresponds to each pixel in all input views. We select the final value of each texel as the median value of all its predictions. This median filter is especially effective at removing erroneous predictions that are not consistent across views (see Sec. 4.7.3.1) including the ones due to re-projection artifacts caused by approximate geometry and camera calibration in real-world scenes.

4.5 Synthetic Training Dataset

We trained our method by generating a dataset of synthetic renderings with corresponding ground truth material maps. We use professional artist-modeled assets in Autodesk 3DS Max with high quality V-Ray materials, that help bridge the gap between training data and real re-topologized scenes. We purchased a set of scenes [1] and extracted basic environments and several different objects that we recombined to create new scene configurations, on which we place objects with different materials.

We augment the initial artist-generated materials with materials from Deschaintre et al. [2018], hand-picked to correspond well with the underlying geometry and to cover a wide range of everyday indoor materials such as wood, metal, plastic, rubber, leather, etc. Each choice of materials and objects provides a scene configuration, for a total of 160 scene configurations, created from 5 "base scenes" with a set of random object placements. We place area and point lights in the scene, as well as environment maps that typically illuminate the scenes through a window.

We rendered each image using a Cook-Torrance BRDF model with a Beckmann normal distribution, which we have implemented in the Mitsuba physically-based path tracer for full global illumination (Jakob [2010]). We subsequently denoised each rendering using the Optix denoiser (Parker et al. [2010]). For each image, we also generate the ground truth SVBRDF maps, i.e., diffuse albedo, roughness and specular albedo rendered as images. Finally, we pre-compute the per-pixel re-projected color statistics (minimum, median and maximum) for each image in our dataset.

[1] From `https://evermotion.org/shop/cat/355/all_scenes/0/0`, Volume 1, 8 and 30.

Figure 4.4: Renderings from our synthetic dataset used for training our network. The dataset has a variety of different lightings, materials and viewpoints.

We render each scene configuration under 40 different viewpoints, yielding a total of 6400 images at resolution 640×384 pixels (see Fig. 4.4 for a small selection). At training time, we extract random crops of 256×256 pixels from each image to be fed to the network, which effectively augments the size of the dataset, to around 45,000 individual crops.

Figure 4.5: Retopologized geometry with selected input images for REAL KITCHEN (left) and REAL HALLWAY (right) scenes.

4.6 Implementation Details

We have implemented our method in python using the PyTorch (Paszke et al. [2019]) framework for deep learning and C++/OpenGL shaders for all steps that require reprojection.

At inference, we run the CNN over images in 640×384 resolution. If the aspect ratio does not match, we zero-pad to fit to the nearest multiple resolution. We expect input images in linear color space, and we apply a log transformation followed by a normalization to $[-1, 1]$ to flatten the dynamic range before processing by the CNN. For real images, we assume gamma correction of 2.2 to convert to linear space.

Our system for dataset generation includes a plug-in for 3DS Max that exports materials into MITSUBA-compatible format using our BRDF model. To handle complex material graphs, we evaluate them in 3DS Max and save texture layers subsequently used by MITSUBA.

We first train the diffuse track and then the specular track separately over 15 epochs each, using ground truth maps for the missing components when evaluating the rendering loss. We then fine-tune the two tracks jointly for 15 epochs. Overall training takes 18 hours on a 4 RTX8000 GPU cluster node. The confidence networks for roughness and specular

maps are trained for 100 iterations. We use the ADAM optimizer (Kingma and Ba [2015]) with a fixed learning rate of $2e-5$ for all training.

We will release all source code for our method, including the dataset generation system, and all training images used for our results.

4.7 Results and Evaluation

We show results and evaluations on two synthetic scenes (Veach Ajar, and Dining Room), and three real captured scene (Real Office, Real Kitchen, and Real Hallway). The capture details and scene lighting conditions are provided in Table 4.1. We also provide comparisons on two additional synthetic scenes (Living Room, and Kitchen). We provide a supplemental video showing view dependent effects over paths and image sequences. We strongly encourage the reader to view the videos to appreciate how our automatically created material maps are directly usable for physically-based rendering and scene editing.

For synthetic scenes, we render a set of views of the scene, and then use these as if they were photographs to run our entire pipeline; we use the original geometry in this case. Note that as a result the ground truth materials are encoded as a single material map texture atlas to be comparable with the results of our method. For real scenes, we take a set of photos of the scene, paying attention to capture highlights in several views, then run structure-from-motion and multi-view stereo to obtain an initial 3D reconstruction. We hired a professional artist to turn these reconstructions into a retopologized mesh (shown in Fig. 4.1(a) and Fig. 4.5). We use Blender automatic UV-unwrapper to unwrap the meshes into texture atlas. The resolution of the texture atlas is 16Kx16K pixels for the scenes we considered.

Synthetic data allows quantitative comparisons on both the material maps and the renderings; for real scenes we can only show qualitative results due to the lack of ground truth maps.

4.7.1 Results

In Fig. 4.6 and 4.7, we show the scene geometry textured with final *approximate* material maps obtained by our method. For synthetic scenes we also show the ground truth maps. Additionally, for both synthetic and real scenes, we show the re-rendered image, i.e., we

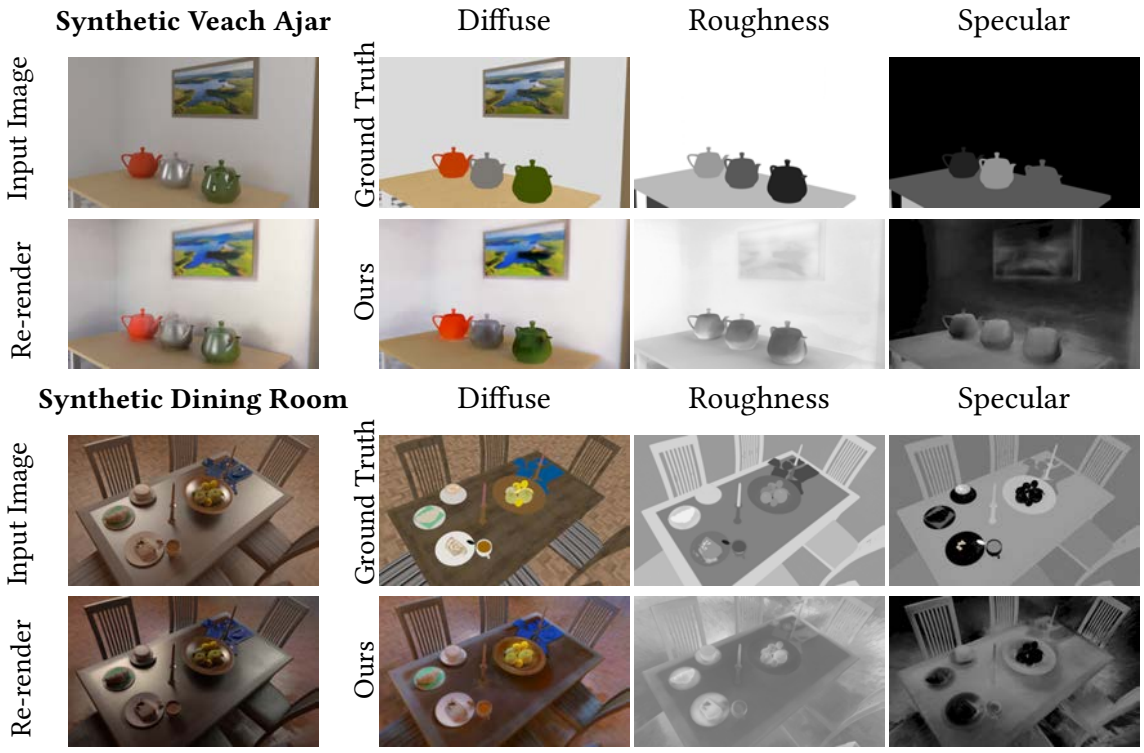


Figure 4.6: Results on two synthetic scenes where the ground truth is available. For each scene, in the first row we show an input view and the ground truth diffuse, roughness and specular maps for that view. The second row shows the re-rendering followed by the maps obtained using our method for the input view shown. We are able to reproduce renderings which are close to the input view using the approximate material maps generated by our method.

generate the material map texture atlases using our method and then provide them to a path tracer along with the geometry to render the scene with full global illumination effects. To achieve a result as close as possible to the input image, we place the lights manually to match input conditions as much as possible. Despite these approximations, our re-renderings are plausible renditions of the input images, illustrating the efficacy of the approximate material maps we obtain.

Our method manages to capture the overall material properties of the objects even in real scenes, e.g., in the Real Office scene the desktop and the red box are shiny while the yellow box on the right and the orange are more diffuse.

In Fig. 4.8 we show results with modified lighting conditions and object insertion from different viewpoints on real scenes. This figure shows that we achieve our goal of creating

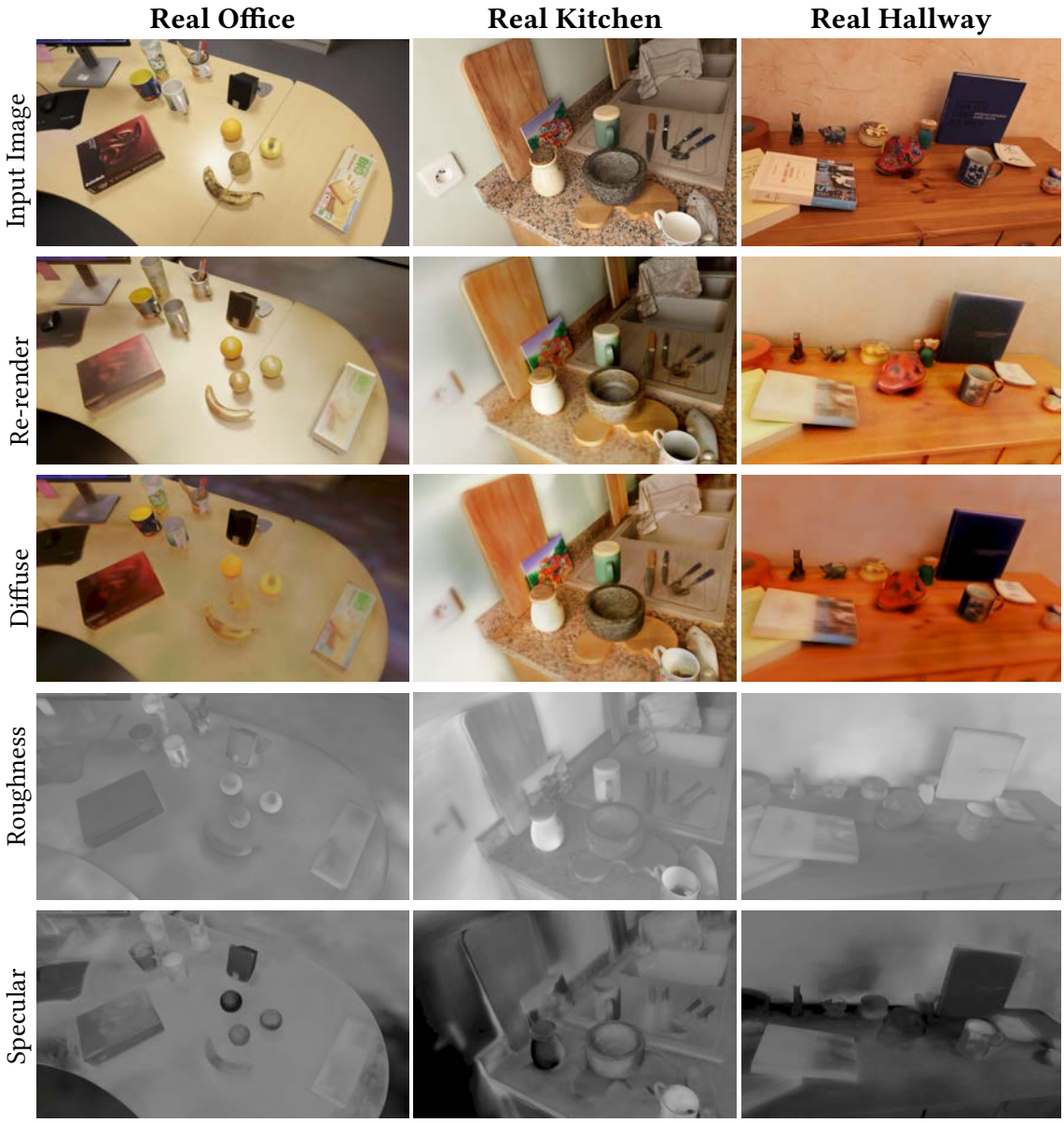


Figure 4.7: Results on real scenes. For each scene, we show the re-rendering in closest matched capture conditions and the maps obtained for the given input view using our method.

plausible material assets for photorealistic scene editing. For each scene, we show a view in input lighting condition and the same view with modified lighting condition. We are able to remove and move shadows and highlights on most surfaces. We further augment the scene by inserting complex objects, such as a metallic statuette in the Real Office

Scenes	#images	Resolution	Pre-processing	Prediction	Texturing ($\times 3$)	Total
Veach Ajar	160	640×384	7.56	3.93	3.12	20.85
Dining Room	105	640×384	4.56	3.26	1.32	9.14
Real Office	245	640×355	11.48	5.84	4.77	31.63
Real Kitchen	296	640×411	32.05	25.25	7.91	81.03
Real Hallway	226	640×414	16.10	10.26	7.03	47.45
Mean			14.35	9.71	4.83	**38.02**

Table 4.1: Timing breakdown for each step of our method on the scenes used for our experiments. All times are reported in **minutes**.

scene and a transparent water goblet and wine bottles in the Real Kitchen scene. Our material maps, together with the retopologized geometry are complete digital assets, and thus allow renderings with full GI interactions between real and virtual objects, such as color bleeding, refraction, caustics and internal reflections when rendered using a path tracer. We emphasize that such effects are not possible to reproduce using relighting based object insertion methods such as Karsch et al. [2011, 2014]; Gardner et al. [2017].

4.7.2 Evaluation

The only other methods designed to handle our input at scene-scale are differentiable rendering approaches (Nimier-David et al. [2021]; Haefner et al. [2021]); unfortunately neither code nor data (in the form of input images we can use for SfM/MVS) is available, precluding direct comparison. In any case, our method can be seen as complementary and could be used as an initialization for these methods, potentially accelerating their process. We report the timings for different steps of our method and compare with timings reported by these previous works to support our claims. Additionally, we present best-effort evaluation using two baselines. We also present a set of ablation studies to analyze the effect of our various design choices.

4.7.2.1 Speed

We show the timing breakdown of each step for our method (pre-processing (Pre-processing), single-view prediction (Prediction), and texture atlas generation (Texturing) on a system with an Intel Xeon Gold 5218 2.30GHz Processor and Quadro RTX 5000 GPU, in Tab. 4.1. We observe that it is possible to create renderable assets from a multi-view dataset with approximately $30\ minutes$ to an hour of computation depending on number

Figure 4.8: Scene editing with our method on real scenes from input and novel viewpoints. From left to right, we show the scene re-rendered with capture configurations, the same view re-rendered with modified lighting, with a virtual object inserted along with modified lights, and the modified scene from a novel viewpoint which is never captured. For example, for the real office scene we have added the statuette; notice how shadows underneath the red book and orange fruit have been (re)moved correctly on modifying lights (first row, blue and red inset) and the color bleeding of the red box and orange on the statuette (first row, green and yellow inset) due to *correct interaction between real and virtual objects* which cannot be achieved by relighting based object-insertion methods. Our method enables insertion of objects with highly complex materials such as in the real kitchen scene, we have inserted a goblet of water and two bottles of wine. Since our method produces digital assets that can be directly used for rendering, our renderings exhibit complex lighting phenomena such as caustics (second row, red inset) and internal reflection (second row, green and yellow inset) which is enabled by our method while removing shadows (second row, blue inset; third row red inset) and correctly predicting the specular material e.g., on the knives and the red shoe-stone in real hallway scene as illustrated by the rendered highlights (second row, red inset; third row, blue inset). Please see the supplementary video for results with moving paths and lights to better appreciate these effects.

of images and their resolution. Previous works dealing with scene-scale material estimation (Nimier-David et al. [2021]; Haefner et al. [2021]) report around $10 - 12$ hours for a complete scene optimization. Thus, our method is able to produce renderable material maps in *a fraction of time* as compared to previous works. Note that the timings reported do not include the time taken for reconstruction and re-topology of the mesh.

4.7.2.2 Comparisons

We compare to two baselines: The first mimics current practice in digital content creation and uses the retopologized geometry to project the input images into texture space, using median filtering to create an RGB atlas (Texture), while the second is based on the single image method of Li et al. [2020] (LiEtAl). Specifically, we run the method of Li and colleagues to generate maps for each input view, then we run the same pipeline as for our method to create a material texture atlas for the scene from the multiple predicted maps. We implement their BRDF model in Mitsuba to generate the re-renderings.

We perform quantitative and qualitative comparisons for our method compared to the two baselines. Since the first baseline does not estimate maps, and LiEtAl infers maps for a different BRDF model, we only compare re-renderings, i.e., we re-render the scene for a set of views. We perform quantitative comparisons by computing PSNR and DSSIM error (Loza et al. [2006]).

We show the numerical results in Table 4.2, and a visual comparison in Fig. 4.9 and Fig. 4.10. For the visual comparison, we show the input view on which the maps are predicted. In line with our aim of generating plausible material assets for scene editing, we generate ground truth re-renderings of a modified scene by rendering the scene with modified lighting using ground truth SVBRDF maps. To show how our method compares against the two baseline for this task, we re-render the scene with modified lighting but with the material maps obtained by the three methods. We see that our approach shows much better variations of appearance, i.e., shiny materials, compared to the LiEtAl that struggles to identify shiny materials and Texture where everything is diffuse by construction. Our method works well for novel views which we show by re-rendering a novel view for each scene. This view was never seen by the network and no maps were predicted for this viewpoint.

The quantitative results are computed on 10 views of each scene selected from a rendered

Method	PSNR ↑			DSSIM ↓		
	LiEtAl	Texture	Ours	LiEtAl	Texture	Ours
Veach Ajar	12.97	18.70	**21.53**	0.37	0.30	**0.27**
Dining Room	21.33	15.12	**21.77**	0.36	0.62	**0.34**
Living Room	13.67	19.97	**25.61**	0.44	0.44	**0.16**
Kitchen	9.06	16.54	**21.28**	0.52	0.42	**0.30**
Mean	14.26	17.58	**22.55**	0.42	0.44	**0.27**

Table 4.2: Quantitative comparison on 4 synthetic scenes. Our method performs best across the scenes which supports our qualitative observations (see also Fig. 4.9 and Fig. 4.10).

path; we also show a video comparison on the rendered path along with the ground truth in the supplemental video. We see in Table 4.2 that our method is numerically best for both synthetic scenes on both PSNR and DSSIM metrics. In Veach Ajar scene our method performs significantly better than the baselines as we can see how our method is able to approximately recover the gradation in specularity between the teapots while both the baselines fails to do so. In Dining Room scene, numerically our results are better yet close to LiEtAl, although it's worth noticing that visually our method is able to do much better. For example, we are able to recover the spatial variation in roughness of the table top which LiEtAl fails to do and ends up over-smoothing the diffuse albedo. This shows the advantage of using multi-view information which helps the network infer the spatially-varying nature of roughness locally as well as globally for each surface point.

Since we do not have ground truth for real scenes, we show qualitative results on Real Office scene in Figure 4.11. We show 5 re-renderings of the scene with modified lighting using our method and the baselines. In line with our observations on synthetic scenes, we observe that our method is able to predict the variations more accurately than LiEtAl which fails to reconstruct highlights on glossy surfaces such as the tabletop and red box. Compared to Texture we can easily see that the surfaces do not reflect the change in lighting and the baked-in shadows and highlights are still present due to the static and diffuse nature of the texture; please see the video to appreciate the visual importance of this effect.

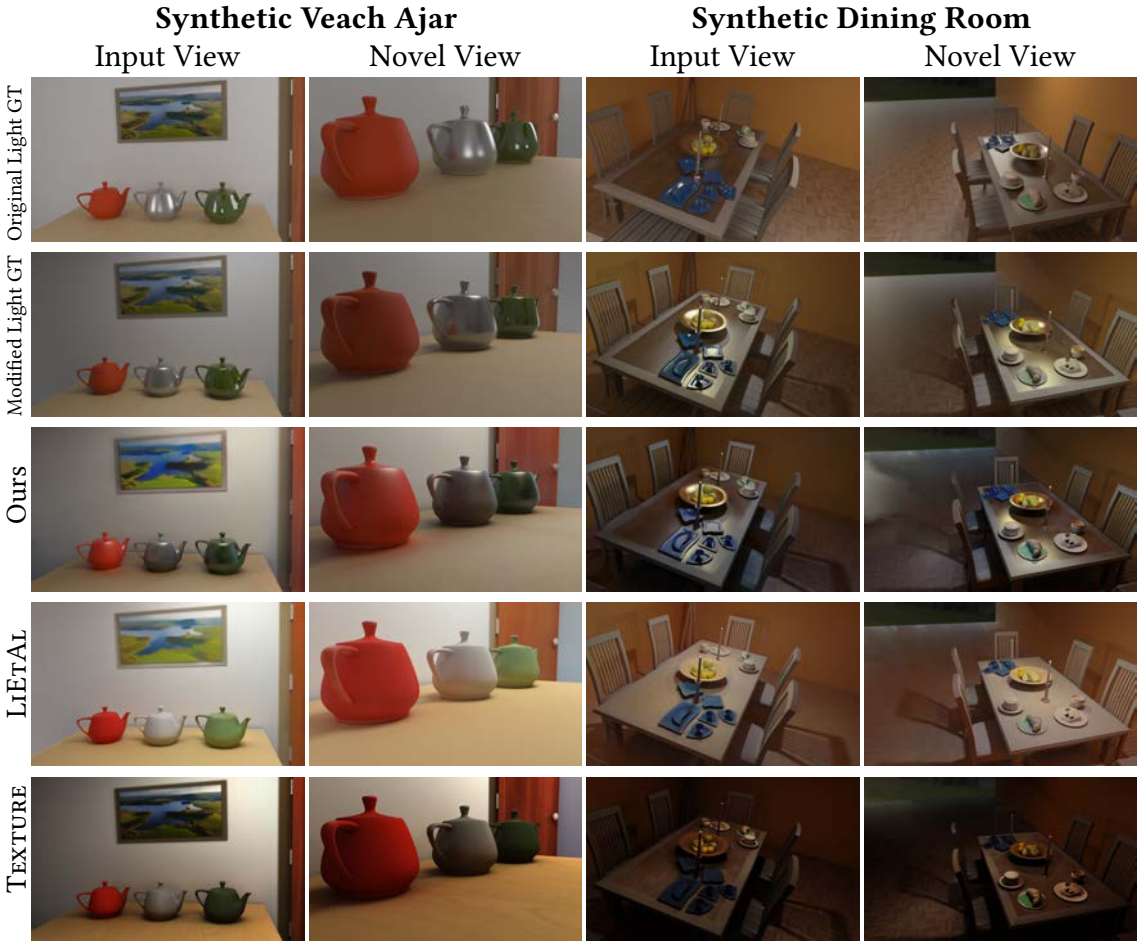


Figure 4.9: Comparisons with baselines on synthetic scenes; For each scene, we show an input view and a novel view; in the first row we show the viewpoint with original input lighting condition (Original Light GT). From second row onwards we modify the lighting in the scene and show the same view re-rendered with modified lighting condition using ground truth maps (Modified Light GT), our maps (Ours), maps produced by LiEtAl and the baseline static Texture generated using the input images. Column 1 and 3 corresponds to an input view while column 2 and 4 corresponds to a novel view for which no maps were predicted. Notice how LiEtAl fails to reconstruct highlight properly and the Texture is composed of pre-baked highlights and shadows from original lighting condition.

4.7.3 Ablations

4.7.3.1 Multi-view Reprojections

We perform a first ablation on the two main components of our algorithm: 1) we remove the statistics reprojected from other views (No RP) and 2) we remove the merging of maps in texture space (Ours Im.). We show quantitative comparisons, where we provide error in form of Mean Squared Error (MSE) for the three maps computed for 10 randomly

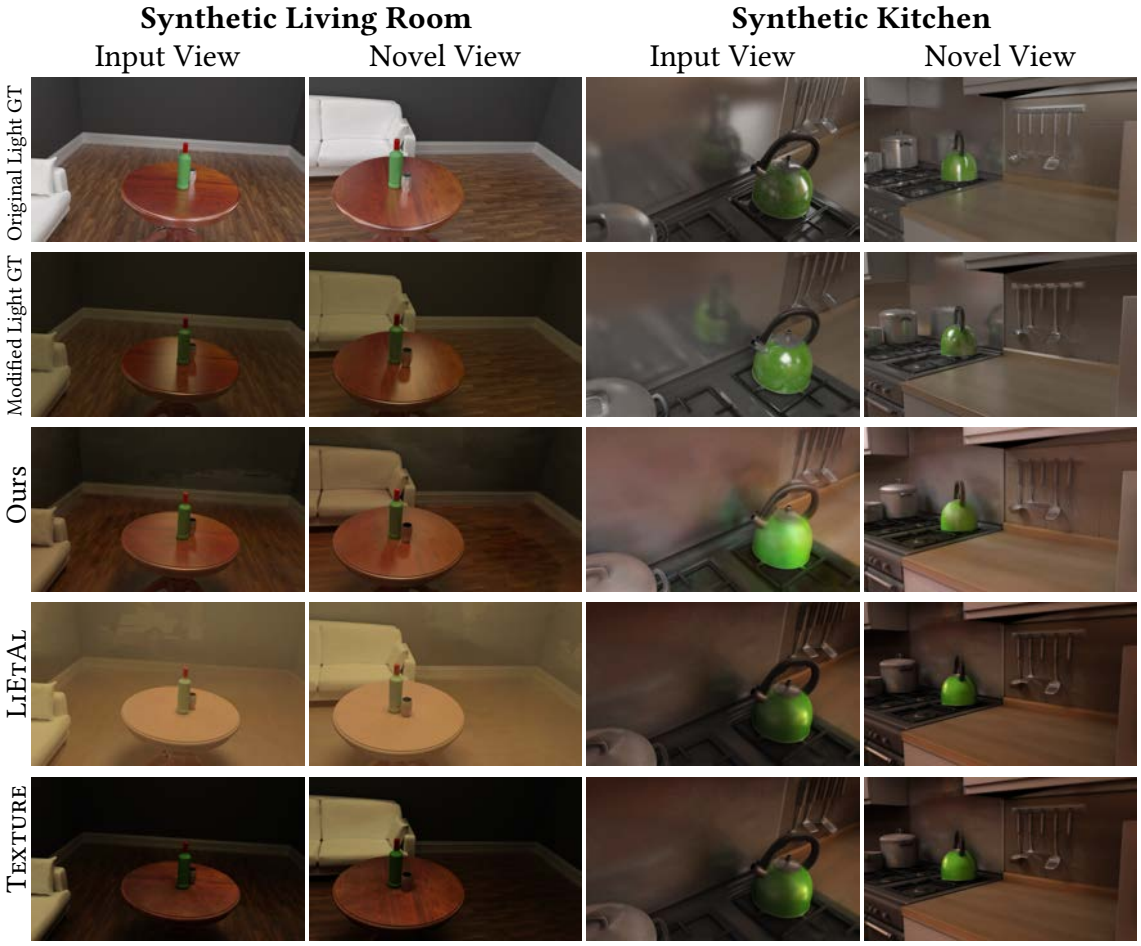


Figure 4.10: More comparisons with baselines on synthetic scenes; The layout is the same as for Fig. 4.9. Similarly as in Fig. 4.9 we see that our approach shows much better variations of appearance, i.e., shiny materials, compared to the LiEtAl that struggles with shiny material sand texture details and Texture that contains pre-baked highlights and shadows from input images owing to its diffuse and static nature.

selected input views. We show quantitative results in Table 4.3,4.4 and an example of the visual effect of the different cases of increasing multi-view information with each step in Fig. 4.12. Using all our components improves results in the majority of cases. The use of the reprojected image statistics makes a very significant difference in the quality of the maps. While merging in texture space may not significantly improve quality, it helps with increasing consistency between different views for the underlying surface material properties (especially for roughness and specular maps) and thus helps improve quality of the final re-rendering. Reprojection improves roughness on most of the objects in the

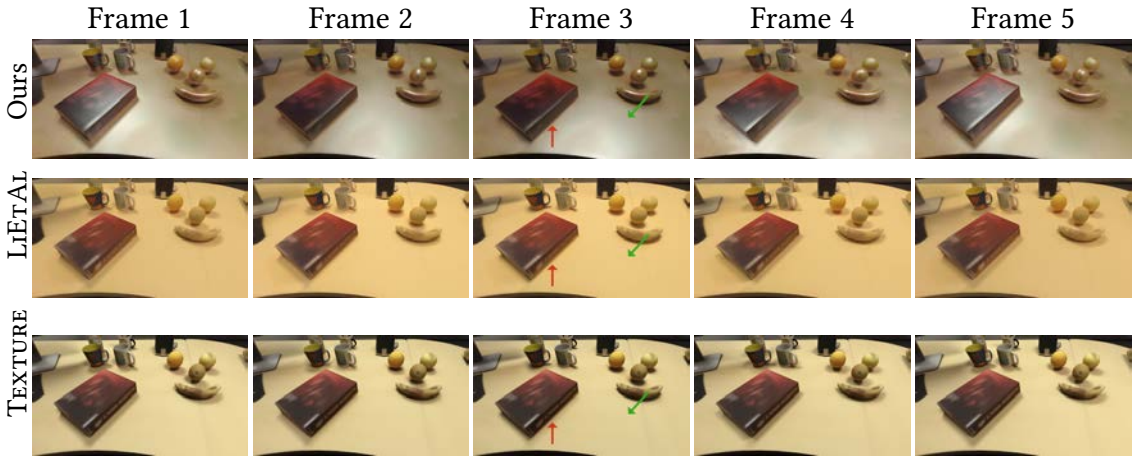


Figure 4.11: Comparison with baselines in Real Office scene. We render a novel viewpoint with changing lighting conditions and show 5 frames with different lighting conditions. Notice how our method (re)move shadows (red arrow) and reconstruct highlights (green arrow) accurately while both LiEtAl and Texture fail to do so. Please see the supplemental video to appreciate the smooth transition of the highlights and shadows as a result of movement of the light sources.

Synthetic Veach Ajar

	MSE ↓		
Method	Diffuse	Roughness	Specular
Ours	**0.022135**	**0.055563**	**0.016522**
Ours Im.	0.030718	0.064362	0.049969
No RP	0.141682	0.372725	0.055459

Synthetic Dining Room

	MSE ↓		
Method	Diffuse	Roughness	Specular
Ours	**0.026491**	0.071609	0.028250
Ours Im.	0.026642	0.077578	**0.024123**
No RP	0.053104	**0.048222**	0.032385

Table 4.3: Quantitative evaluation for the ablation on the synthetic scenes on the material maps, with 1) no reprojected statistics, 2) no multi-view merge in texture space. We see an increase in multi-view information helps improve the maps quality and/or consistency across views (see also Tab. 4.4 and Fig. 4.12).

scene, but sometimes has a negative effect on background parts that lack observations (see Fig. 4.12); this explains why No-RP has better MSE for roughness in Dining Room in Table 4.3.

Method	Veach Ajar		Dining Room	
	PSNR ↑	DSSIM ↓	PSNR ↑	DSSIM ↓
Ours	**19.520969**	**0.190185**	**19.35363**	**0.244253**
No RP	11.338421	0.413321	17.675568	0.343187

Table 4.4: Quantitative evaluation for the ablation on the synthetic scenes on the re-renderings, with 1) no reprojected statistics, 2) no multi-view merge in texture space. Using multi-view statistics achieves better re-rendering quality than using only a single view (see also Tab. 4.3 and Fig. 4.12).

4.7.3.2 Inaccurate Geometry

We study the robustness of our method for inaccurate geometry by running our pipeline on the degraded mesh obtained directly from multi-view stereo (MVS) (Reality [2018]) which consists of large holes and bumpy surfaces. The bumpy surfaces are an instance of extreme vertex perturbation. We show a qualitative comparison in Fig. 4.13. From the figure we can confirm that the maps obtained from the MVS mesh is only slightly degraded compared to the re-topologized mesh. Thus, our material estimation is robust to geometrical inaccuracies. While the maps obtained are similar, the final image obtained after re-rendering is highly degraded when rendered since the MVS geometry has bumps and holes. To obtain high quality re-renderings we need good geometry, justifying our design choice of using re-topology. In future work, it may be possible to adapt previous methods (e.g., Yu and Lafarge [2022]; Bauchet and Lafarge [2020]) to provide geometry that corrects these errors for flat surfaces, but it would be necessary to preserve the relatively well reconstructed irregular objects (such as the fruit on the table).

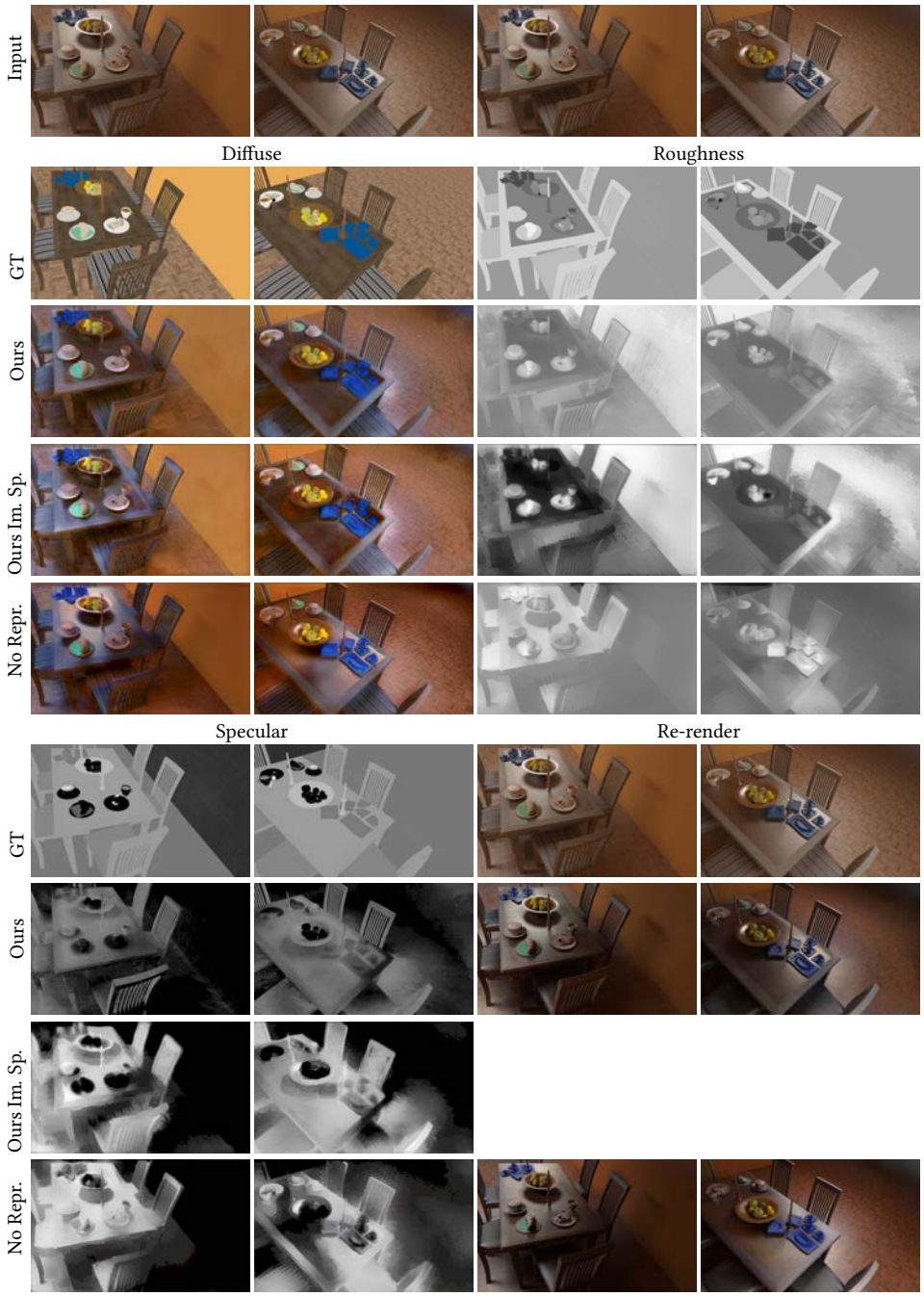


Figure 4.12: Example images from Synthetic Dining Room showing the effect of increasing multi-view information on results. Note how the quality of the maps is significantly improved by using the reprojected statistics observed in our image space predicted maps as compared to no reprojection, i.e. using only a single image. Furthermore, gathering the image space maps in texture space helps improve the consistency of the maps across views and thus improves re-rendering by assigning same material in local regions (esp. in roughness and specular maps). As a result, re-rendering is closer to ground truth.

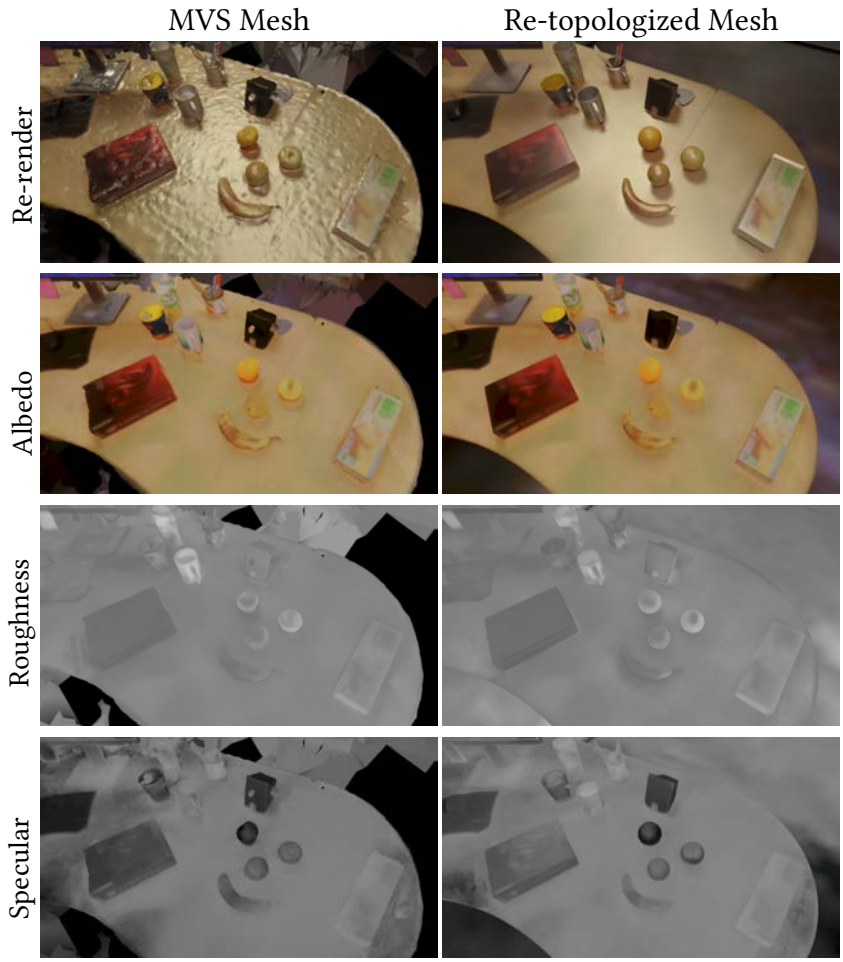
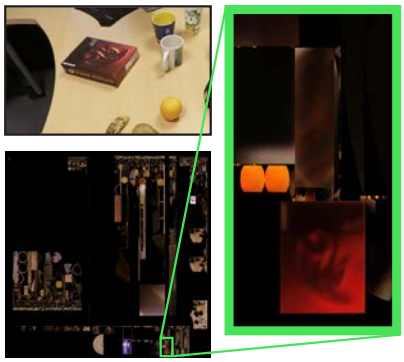


Figure 4.13: Effect of inaccurate geometry. We run our pipeline on the MVS mesh obtained directly from Reality [2018]. The results from the MVS mesh is on the left column and our re-topologized mesh on the right column. We show a re-rendering in the first row, followed by the albedo, roughness and specular maps obtained using our method in subsequent rows respectively.

4.8 Limitations and Future Work



Despite yielding convincing re-renderings on synthetic and on real scenes, our method still has several limitations. Our final texture-space maps often suffer from limited resolution, since the texture atlas can provide only limited space for a given object (see inset, where the texture of the red box covers only a very small part of the texture atlas). This is an inherent problem with texture-space methods, and alternative approaches (e.g., see Yuksel et al. [2019]) could be a good direction for future work. While reprojection error may be a contributing factor for blurriness in results, we believe our use of median filter to merge and obtain texture-space maps alleviates this problem. We focused on estimating BRDF parameters for each texel of a texture atlas. A natural extension would be to also estimate per-texel normals expressed in a local coordinate frame, which would allow the reproduction of small geometric details not modeled in the retopologized mesh. However, such small-scale relief is often difficult to perceive when captured from far away.

In some cases, e.g., the banana and pear, our method predicts glossiness that is high; we hypothesize that this is due to the re-projection errors due to the mismatch between the re-topologized and real scene geometry. The effect of this is visible in the reprojected min image (see inset).

The dataset we created to demonstrate our approach offers limited variability, which in turns limits the ability of our method to handle diverse scenes. While we provide our toolbox to generate additional training images, rendering large datasets is costly and could benefit from strategies to reuse computation across view (Fraboni et al. [2019]); we hypothesize that augmenting the variety and the number of training images seen by our network will improve results overall, possibly helping remove shadow and incorrect color residuals that are sometimes still present in our albedo maps.

We rely on fixed color statistics to aggregate the multi-view information that we feed to our per-view CNNs, and we employ a fixed median filter to merge the resulting per-view predictions into texture space. Replacing these two operations by differentiable

pooling in a learned feature space could yield improved predictions, as has been done in other applications (Su et al. [2015]; Kalogerakis et al. [2017]). However, training such an architecture end-to-end raises specific challenges, such as storing multiple CNN tracks in memory and performing differentiable re-projection in the texture atlas while doing per-image processing.

4.9 Conclusion

We have presented the first attempt at creating scene-scale material map textures of indoor environments using deep learning. Our solution retains the strength of image-space CNNs, which have proven successful at recovering material parameters for close-up photographs of flat surfaces and isolated objects. To apply such CNNs at scene scale, we first inject multi-view information by computing statistics about a pixel color when re-projected into neighboring views. We then move to texture space to merge per-view predictions into a single texture atlas suitable for rendering.

Our method allows automatic generation of material maps that allow plausible renderings, retaining the overall look of multiple objects in a scene, both for synthetic scenes with available ground truth and for real scenes. Our results demonstrate that by exploiting reprojected multi-view data, improving the rendering loss and exploiting state-of-the-art components it is possible to provide an operational pipeline to extract convincing materials at scene-scale from a set of images as input.

In the method we presented in this chapter, we used a synthetic dataset rendered with a traditional path tracer (Jakob [2010]) to train our network. Learning-based methods required large amounts and variability of data. We noted that the dataset we used provided limited variability and we need to augment the dataset in order to expose the network to wider variations in scene conditions. The limitation arises due to the use of traditional rendering which is slow and could not generate a wide variety of scenes despite our best efforts to maintain variability in the scene conditions used to render the dataset. Additionally, data storage and transfer also take significant amount of time for large datasets required for learning-based tasks. To alleviate these problems, we look at neural rendering approaches which have recently been shown to learn neural scene representations for rendering a scene with significant variability. Our neural approach discussed in the next chapter generates images at a significantly faster rate than

traditional rendering and can produce images showing wide variety in the parameters we seek to vary for on-the-fly training of neural networks.

Chapter 5

Synthetic dataset generation using neural rendering

In the last chapter we used a learning-based method to estimate material properties of real captured scenes. To train the network we generated a new scene-scale synthetic dataset using the Mitsuba renderer (Jakob [2010]). As reported in Sec. 4.5 we generated 6400 images of scenes with significant variations, which took 24 hours to render with sufficient quality. Despite our best efforts we observed some incorrect residual in the predicted albedo maps which may be a direct consequence of limited variability in our dataset. Thus we want to generate more data but using traditional rendering method will be slow. In addition, such a process would require substantial storage capacity making it difficult to generate large quantities of data.

To speed up data generation and increase variability we explore neural rendering as an alternative to path tracing. Forward rendering with neural rendering is recently emerging as a new approach of rendering scenes with variations (Diolatzis et al. [2022]). Due to its feature of producing images with full global illumination at a rate significantly faster than path tracing, neural rendering is highly suitable for on-the-fly data generation for training tasks. Using neural rendering, generating variations of a given scene by modifying its parameters is a highly flexible method to generate the type of data required by the learning task, potentially speeding the learning process. We study the pros and cons of using neural rendering for data generation in this chapter, highlighting different scenarios where neural rendering has clear advantage over path tracing and identifying some challenges to overcome in the process.

5.1 Introduction

Computer graphics in recent years has turned towards deep learning to develop many state-of-the-art methods (Keller et al. [2018]; Glassner [2019]). Supervised learning is

a branch of deep learning which trains a neural network using sets of paired data to learn a set of filters which can reproduce the result on unseen data. For many tasks the success of deep learning algorithms depends on the quantity and variability of the data seen by the network during training. Deep neural networks need large quantities of data to be able to learn a task efficiently. Obtaining real captured data with annotations in such quantities can be impractical for many tasks. Using synthetic data to train a neural network for task which specifically require visual data is very common now a days (Li and Snavely [2018]; Li et al. [2020]). However, generating highly photorealistic dataset using traditional rendering algorithms in such large quantities to train a deep network has several issues.

Quantity, variability and realism are the main requirements of a deep learning dataset. Realism is quite often achieved by using traditional rendering approaches specifically using a path tracer to render. While traditional rendering helps with creating photorealistic images, reducing the domain gap between real and synthetic images in the process, it is difficult to create them in large quantities. Traditional rendering using path tracing is slow and can require several minutes or hours to render an image with sufficient quality. Since the requirements of deep learning task can be well into hundreds of thousands of images, rendering them with path tracers may require days or even months and can quickly become infeasible.

Storage of large dataset is another issue specially with visual data which typically range between hundreds of kilobytes to a few megabytes per image. To further compound the problem, I/O overhead of reading and writing such large quantities of images while training also adds to the already large cost of rendering them. In terms of variability, traditional rendering algorithms can render datasets with significant variations, but to achieve that they require wide variety in 3D scene assets which can be hard to obtain. 3D assets for traditional rendering such as meshes and textures are mostly generated manually by 3D artists making it a difficult task to obtain and assemble them for rendering large dataset.

Neural rendering is emerging as an alternative to traditional rendering. While a number of recent works involving neural rendering focus on inverse rendering, there are some works which focus on learning a neural scene representation to achieve forward rendering with global illumination. Amongst these Diolatzis et al. [2022] have features which make

them suitable for dataset generation tasks. First, the method is faster than traditional rendering approaches, rendering several images per second with global illumination. Second, the neural scene representation learned by this method is capable of varying scene parameters such as geometry, materials, lighting, viewpoints, positions etc. which is of prime importance for several learning-based tasks. Finally, generating a training dataset on-the-fly solves the problem of storage and I/O of large dataset requiring no overhead with the above mentioned benefits.

We study the feasibility of using neural rendering for synthetic data generation instead of traditional rendering methods. For this purpose we compare two recent rendering methods, *Mitsuba 2* (Nimier-David et al. [2019]) for path tracing and *Active Exploration (AE)* (Diolatzis et al. [2022]) for neural rendering. We first compare the two solutions in terms of speed of generating images of comparable quality. While Mitsuba 2 can start generating images as soon as the scene description is ready, AE needs a few hours to learn the scene representation before it can generate images. Once the scene representation is learned, the network can render images at a much faster rate as compared to Mitsuba 2, quickly catching up with it.

A more interesting scenario arises when we compare the two data generation methods by allocating the same total computational budget for training and data generation. We divide the total computation budget into *dataset generation time* and *training time* and study different configurations of budget allocation for both rendering methods. We learn that while Mitsuba 2 initially generates more images because of the overhead of learning the neural scene representation for AE, AE is able to generate data at a much faster rate thus quickly overcoming Mitsuba 2 while training the network simultaneously on-the-fly. As a result, AE presents a better overall balance between data generation and training tasks. We setup our system to generate images of a scene with Mitsuba 2 and to learn the same scene with similar variations using AE. We use the data generated using the two methods to learn the task of intrinsic image decomposition with the network we used in Chapter 4 and provide results of initial experiments for learning the given task.

On-the-fly data generation for training presents opportunities to generate images which contributes only relevant information in training the network. Researchers have studied generating relevant samples from the space of all possible samples to accelerate learning tasks with fewer samples using *active learning* (Settles [2009]). Once we start generating

synthetic samples using neural rendering on-the-fly, we aim to use active exploration again to try and generate samples which maximize the information provided to the network. The goal of the second exploration strategy will be to aid the learning task by generating only relevant samples such that the learning becomes faster and use fewer samples than the vanilla training strategy.

In summary, our contributions are:

- We show that neural rendering is faster and can generate significantly more images of the same quality as a traditional renderer in the same time.
- On the task of learning intrinsic image decomposition based on a previous method we study the trade-offs between using traditional rendering with Mitsuba 2 and neural rendering with Active Exploration for dataset generation and training.

5.2 Related Work

We discuss some relevant scene-scale synthetic datasets for learning-based tasks, the systems used to generate them and summarize how they contrast with neural rendering. We also provide a brief overview of active learning methods in our active exploration context.

5.2.1 Traditional Dataset

One of the first scene-scale synthetic dataset was SUNCG proposed by Song et al. [2017] for the joint task of scene completion and semantic annotation. The dataset contained more than $45,000$ synthetically generated indoor scenes with depth and semantic labels. For the more involved task of intrinsic image decomposition Li and Snavely [2018] created a custom dataset, CGIntrinsics. The dataset provided over $20,000$ images of synthetic scenes with labels decomposed into reflectance and shading. They used the Mitsuba renderer (Jakob [2010]) to render their scenes and labels which took around 6 *months* on a cluster of 10 machines taking approximately 30 minutes per image as reported. Recently, Meta Reality Labs came up with the Replica dataset (Straub et al. [2019]) which provides high quality reconstructions of common indoor spaces. They provide the geometry and textures along with semantic and instance segmentation as labels. While the CGIntrinsics dataset is available for academic use, other datasets such as the Replica dataset and SUNCG are proprietary or only partially available for further research. In contrast,

current neural rendering algorithms are open-source and can generate synthetic images of similar quality at a much faster rate of several images per second.

More recently Li et al. [2020, 2021] created the OpenRooms synthetic dataset for indoor scenes which used physically-based materials and lighting to generate highly photorealistic images. They provide several different labels for inverse rendering, viz. albedo, roughness, normal, depth, semantic and instance segmentation. While the OpenRooms dataset attempts at being versatile to be used for wide variety of learning tasks involving indoor scenes by providing different labels, each learning task is different and requires different sets of labels. For example the material model used by Li et al. [2021] to render the images were not suitable for our material estimation task in Chapter 4 which involved a different material model. Using neural rendering to generate datasets gives the freedom to the user to customize the labels generated based on the task the dataset is generated for.

To address the requirement of custom labels for different learning tasks, researchers have recently started to create dataset generators which can provide on-demand images and labels. The Hypersim dataset (Roberts et al. [2021]) provides highly photorealistic images for the task of semantic scene-understanding. The dataset consists of over $77,000$ images of 461 indoor scenes providing labels for the geometry, materials and lighting per-pixel. The authors also provide the Hypersim Toolkit which can be used to generate custom images and label as per-requirement and render them using their cloud-based renderer. While the dataset solves the problem of obtaining variable images and labels as per-requirement, the nature of the labels are fixed and a different requirement can not be met since the renderer is not accessible. A new dataset generation pipeline Kubric (Greff et al. [2022]) is under development which is built over pybullet (for physics simulations) and Blender (for rendering). The code promises to be modular to offer flexibility towards rendering backends and can be developed to address requirements for obtaining custom labels as per users requirements. Neural rendering offers this flexibility as well with its plug-and-play nature without going through the overhead of setting up large systems for dataset generation.

5.2.2 Active Learning

On-the-fly data generation and training sample reuse has been studied in the branch of machine learning which deals with active learning. The aim of active learning is

to use as few samples as possible for training. Thus it can reduce data generation cost by procedurally generating data which contributes maximum information to the training. Settles [2009] provides a nice review of active learning methods which decides when a data sample needs to be labeled. Demonstrated in both supervised learning with CNNs (Sener and Savarese [2017]) and unsupervised learning using generative adversarial networks (GANs) (Zhu and Bento [2017]), different active learning principles can be used to improve training sample generation. Diolatzis et al. [2022] introduced Active Exploration which identifies hard samples to train their generator network and even proposes mutations to identify harder samples to overcome overfitting. We aim to modify the active exploration strategy used in Diolatzis et al. [2022] to better suit the training task in order to generate the images using neural rendering having configurations which maximizes the contribution in training.

5.3 Neural Rendering for Data Generation

To discuss data generation with neural rendering we consider different neural rendering approaches available to us. We describe the approach by Diolatzis et al. [2022] in detail highlighting the advantage of using this renderer. We further describe additions to the renderer which we implemented to improve the data generation process.

5.3.1 Neural Rendering

In Sec. 2.4 we discussed several recent neural rendering methods which learn neural scene representations to generate new images of the scene. These scene representations can be learnt from previously rendered images of the scene with or without auxiliary buffers describing the scene content. Eslami et al. [2018] feeds multiple images of a scene to a Generative Query Network (GQN) which is able to learn the scene representation in a latent space. The user can query a vector from this latent space to render an image of the scene with each vector representing a different scene configuration such as a different viewpoint. The idea was advanced by Granskog et al. [2020] who introduced a set of G-buffers to aid the network disentangle between individual scene components such as the geometry, materials and lighting. This helped interpret the neural scene representation learnt by the network which in return provided explicit control over individual scene elements simplifying scene manipulation. Nonetheless they still require upto three full path traced images to make a new observation.

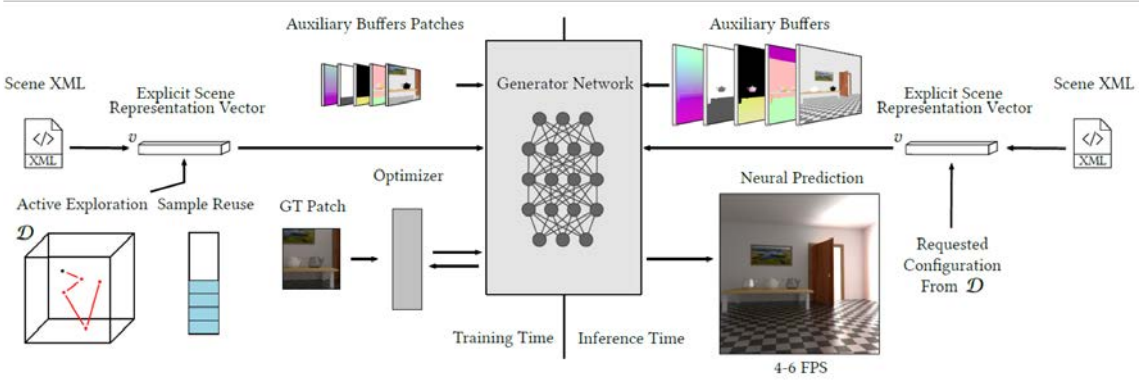


Figure 5.1: Overview of the method proposed by Diolatzis et al. [2022] which has a training phase in which the generator network is trained using ground truth samples generated using path tracing and an inference phase which generates novel configurations of the scene by querying the generator network using the explicit scene representation vector as desired by the user at interactive frame rates. (Figure from Diolatzis et al. [2022].)

One problem with Eslami et al. [2018] and Granskog et al. [2020] is that their scene representations are in a latent space which is not easily interpretable by the user. To remedy this Diolatzis et al. [2022] introduced an explicit scene representation vector v (see Fig. 5.1) which has physical meaning for the users. Fig. 5.2 (left) shows a simple scene configuration with the scene representation vector v of the Cornell Box. The variable parameters are the albedo of the left and right walls and the positions of the two boxes and the light source. Each variable parameter holds the normalized parameter value and is concatenated to form the scene representation vector. Each variation of the scene representation vector corresponds to a single scene configuration and the space $\mathcal{D}$ composed by the scene representation vector defines all possible variations of the scene as shown in Fig. 5.2 (right). This explicit scene representation helps the interpretability and editability of the scene which can be modified to get new scene configurations as desired by the users.

To generate images with variable scene configurations, initially the generator network is trained using ground truth samples generated via a path tracer corresponding to specific scene configurations during a training phase (see Fig. 5.1). We also pass a set of auxiliary scene buffers to the network which stores information about the scene properties in the given configuration. These buffers aim to provide some information regarding the scene to the network that a path tracer would require to solve the rendering equation. More

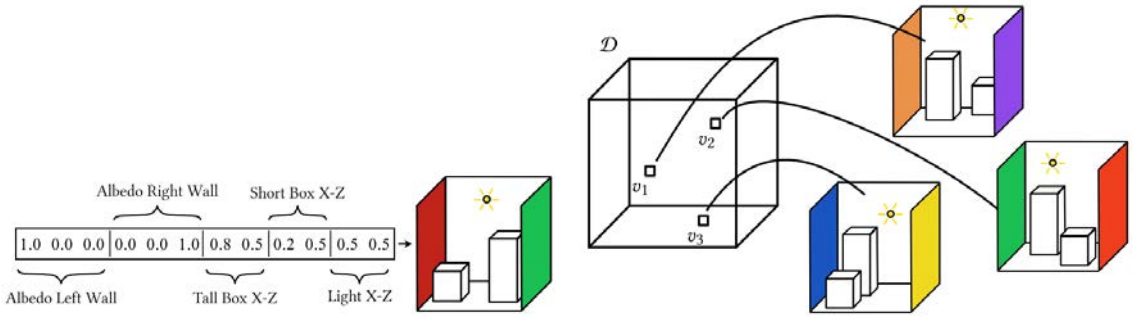


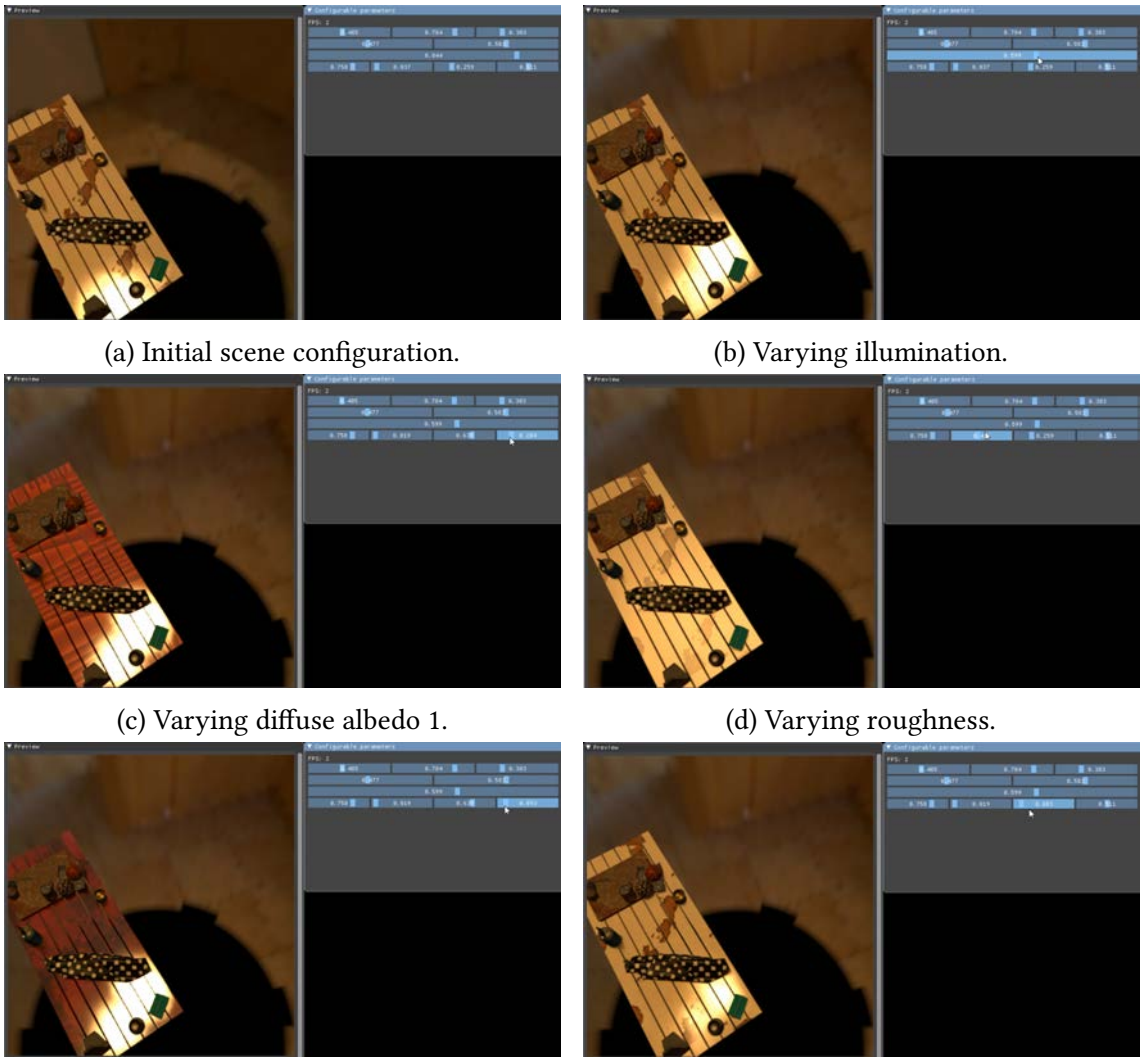
Figure 5.2: Diolatzis et al. [2022] uses an explicit scene vector v (left) which stores the normalized parameter value and can be meaningfully edited to reach a different scene configuration in the sample space of all variable scenes $\mathcal{D}$ (right). (Figures from Diolatzis et al. [2022].)

specifically, we provide the first intersection G-buffers storing the positions and normals for geometry, reflectance and roughness for the BRDF and outgoing direction at each visible surface point of the scene. We also provide the network with emission buffers to give information about the illumination of the scene. Once the network is trained, we can query the network with the desired scene representation vector along with the pre-computed but inexpensive G-buffers to render at interactive frame rates of 4-6 FPS. Due to its speed and ease of generating variable scene configurations, the renderer is suitable to be used for on-the-fly data generation for training another network.

5.3.2 Extensions to Diolatzis et al. [2022]

The initial implementation of the neural renderer by Diolatzis et al. [2022] supported a range of variations including positions of objects, light sources and sensor in a scene, color and intensity of emission, and the reflectance of diffuse and roughness of glossy materials. We retain all variations implemented and add some additional variations to improve the effect of using more complex BSDFs on surface of the objects.

We implemented the Cook-Torrance material model with Beckmann distribution as described in Chapter 4 in Mitsuba 2 (Nimier-David et al. [2019]) and made the diffuse, roughness, and specular term variable. As a result we support roughness with diffuse materials as well to recreate surfaces like smooth plastic and rough wood. The specular term represents the specular strength of the highlight which controls the footprint of the specular lobe on the surface. While the texture variations was only implemented



(a) Initial scene configuration.

(b) Varying illumination.

(c) Varying diffuse albedo 1.

(d) Varying roughness.

(e) Varying diffuse albedo 2.

(f) Varying specular strength.

Figure 5.3: Variations supported by active exploration. We implement additional variables to support complex material variations such as diffuse textures, specular strength and roughness.

for diffuse surfaces, we extended it to support glossy and specular surfaces as well to have more textured and shiny surfaces mimicking materials such as leather, marble and plywood. We show a subset of these implemented variations in Fig. 5.3.

Adding the above mentioned variations makes learning the scene using the generator network harder. As a result the training time increases for scenes with multiple variations and especially with many variable materials. Once we train the network efficiently, our

learned network can generate images with materials that vary from diffuse to glossy and highly specular with textures from a single model.

5.4 Training with on-the-fly Data Generation

We now study the feasibility of using the neural renderer for on-the-fly data generation. We compare the rendering rates of the neural renderer by Diolatzis et al. [2022] with a recent traditional path tracer Mitsuba 2 (Nimier-David et al. [2019]). Then we look at scenarios were neural rendering provides an advantage over traditional rendering for a specific training task and report results of our initial experiments.

5.4.1 Mitsuba 2 vs. Neural Rendering

Mitsuba 2 (Nimier-David et al. [2019]) is a modern renderer which uses GPU-acceleration for path tracing. Thus, of all traditional renderers available, Mitsuba 2 stands as one of the fastest path tracers available to us. For a fair comparison of the rendering rates, we keep the rendering resolution fixed at 256×256 for both the neural renderer and Mitsuba 2. The rendering rate for the path tracer also depends on the number of samples used per pixel, which also affects the rendering quality. High sample count makes the rendering slow but improves the image quality while providing low sample count is fast but introduces noise in rendered images. We give a moderately high sample count of 512 for path tracing to render low noise images of similar quality (MSE: 0.02-0.04) as obtained using neural rendering.

Fig. 5.4 shows the plots which compare the rendering rates of the two renderer for two different scenes. As we can see from both plots, Mitsuba renders at a slow rate generating images constantly with time. Neural rendering takes a variable amount of start-up time for training the generator network for a given variable scene after which we can start rendering images. The start-up time for each model depends on the scene complexity and is usually around $5 - 10$ hours on a single NVIDIA RTX 6000, as reported in the original paper (Diolatzis et al. [2022]). For the scenes we consider for this comparison, the network took 5 hours to train on the same hardware.

Once trained, we start generating images at a much faster rate of 3 images per second using the neural renderer on the first scene (see Fig. 5.4 (left)). Mitusba 2 on the same scene is able to render at a rate of 2.25 seconds per image which gives a maximum of

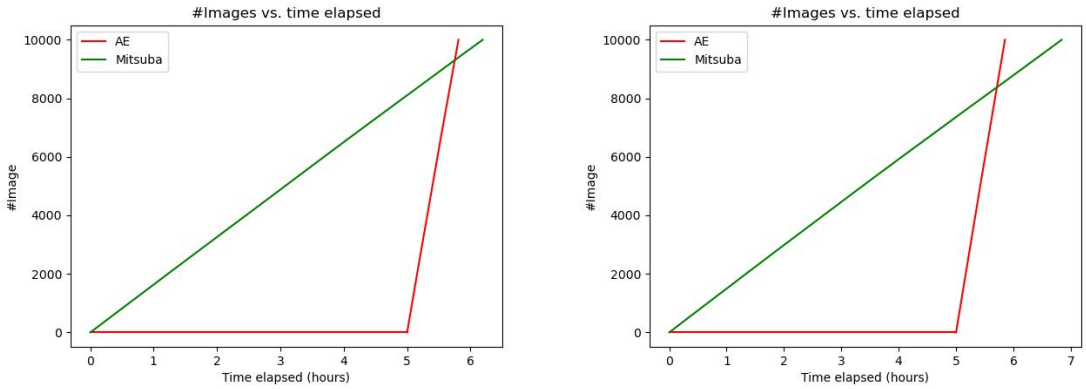


Figure 5.4: Plot of rendering rate of Mitsuba 2 as compared to Active Exploration (AE) for 2 different scenes. AE has an overhead of learning the scene representations, but quickly catches up with Mitsuba 2.

$6-7$ times speedup using neural rendering for the current scene. At this rate, neural rendering is able to catch-up with Mitusba 2 after less than 1 hour of generating images and we start obtaining the same number of images at the 6 hour mark or after about 9000 images. Thus, if we generate images for more than 6 hours or if we need more than 9000 images for training, which is quite low for a typical training task, we will generate more images in less time using neural rendering than using Mitsuba 2. We observe similar behaviour on the second scene as well (see Fig. 5.4 (right)).

5.4.2 Intrinisic Image Decomposition

We now compare data generation with training a network for another task jointly. We chose the task of intrinsic image decomposition (Li and Snavely [2018]; Meka et al. [2018]) for training. The goal of intrinsic image decomposition is to factor a given image into its diffuse reflectance and shading by predicting the diffuse reflectance. The shading (S) can be deduced from the diffuse reflectance (D) by dividing it from the image (I), i.e. $S = I/D$. We use the diffuse albedo prediction network from our last chapter (Sec. 4.4.2) and feed it a single image to predict the diffuse reflectance of the given image.

To compare data generation with training for the two rendering methods, we allocate a *total computational budget* to both methods. The total computational budget can be divided into *data generation time* and *training time*. To compare the performance of both methods, we can compare a) number of images generated and b) the performance of

the network on the training task for each method at the end of the total computational budget.

Using Mitsuba 2 for data generation and training is straightforward. In traditional training we first generate data and then train the network. Based on the budget allocation for data generation and training we can have multiple scenarios for Mitsuba 2. We can allocate a low budget for data generation and have multiple epochs of the same data during training phase. Iterating multiple times through the same images may limit new information provided to the network. Alternatively, we can go for a maximum image generation strategy but that would compromise on training time.

We allocate a fixed percentage of total computation budget for data generation and use the rest of the budget for training. For fair comparison, we go for a relatively even split of data generation time and training time. We start with 60% of the total time for data generation and the rest for training. We allocate a total of $10\ hours$ as our total computational budget. Going by the above split, we give $6\ hours$ for data generation with Mitsuba 2 and the remaining $4\ hours$ for training. Fig. 5.5(dashed green plot) shows the plot of number of images generated with time for the simple case of training with Mitsuba 2 generated images.

On the other hand, using a neural renderer for training is different from traditional training. We are generating data on-the-fly and the data generation is interleaved with training. Thus separation of data generation time and training time is not clear for this case. But we do know from Fig. 5.4 that a) neural rendering has a much faster rate of generating data than Mitsuba 2, and b) there is an initial start-up time in which the neural renderer learns the variable scene for data generation. We include the scene learning time in data generation time, and allocate the remaining time for training with on-the-fly data generation as shown in Fig. 5.5.

Since the rate of rendering images with the neural renderer is fast, there will come a time when the renderer will be able to exceed the number of images generated as compared to Mitsuba 2. But since some time is taken in training as well, it will be at a later stage than when only generating data. Fig. 5.5 (solid blue plot) shows this scenario as a plot of number of images generated with time elapsed. For the simple scenario we considered allocating a total computational budget of $10\ hours$, $5\ hours$ is allocated to learning the neural scene representation model, and the remaining $5\ hours$ is used for training with

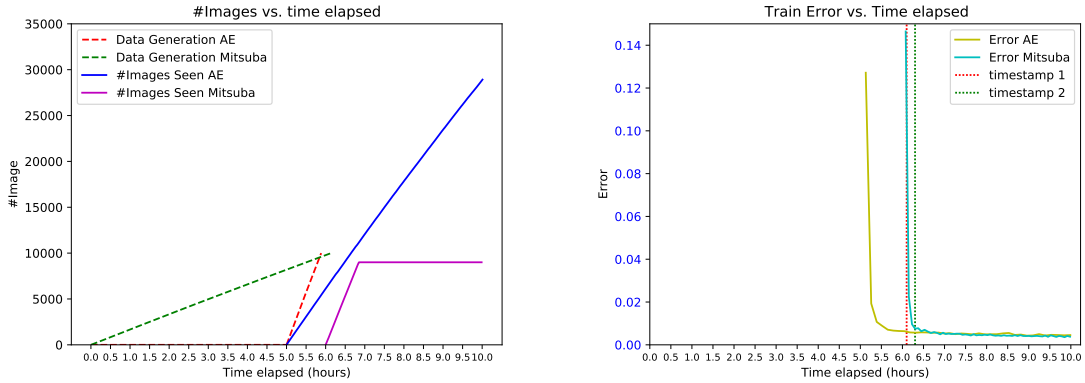
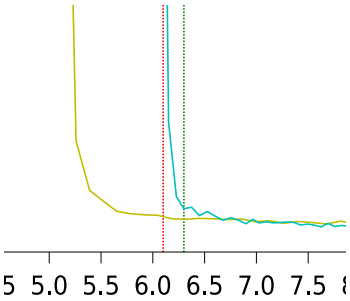


Figure 5.5: Plot of data generation with training over a total computational budget of 10 hours. On the left we compare number of images generated (dashed plots) and number of unique images seen by the network once training starts (solid plots). On the right we show training error for the two models since the start of training for each. Traditional rendering with Mitsuba 2 (dashed green) (left) starts generating images immediately but neural rendering with Active Exploration (AE) (dashed red) (left) quickly catches up and overshoots Mitsuba 2. In practice we generate data on-the-fly for training with the neural renderer (solid blue) (left) so it takes more time to generate data but the network already starts training (after 5 hrs.) as seen by decrease in training error (yellow plot) (right). Mitsuba 2 in comparison starts training at 6hrs. and the error (cyan plot) (right) is much higher than training with neural renderer. We provide results of two timestamps (dotted red and green) (right) where training with AE has lower error than Mitsuba 2 and the network has seen more images with AE than Mitsuba 2 as well.

on-the-fly data generation. We can also observe from the plot in Fig. 5.5 that at the end of total time, the network is able to see many more unique images using neural renderer (approx. $30,000$) as compared to traditional renderer (9000).

5.4.3 Initial Results



In Fig. 5.6 we show initial results of training the intrinisic image decomposition network using data generated from the neural renderer on-the-fly compared with the traditional renderer. We show snapshot of training progress at two timestamps at $6.1hrs.$ and $6.3hrs.$ for both training. We observed from the plot in Fig. 5.5 (right) and as shown in inset (right) that the train error is lower for the neural renderer (AE) than traditional training using Mitsuba 2 generated images. This is confirmed visually in Fig. 5.6 as the prediction is much closer to ground truth for training with neural renderer than traditional renderer at each timestamp for the two input images. Eventually both the network converges to the correct prediction but the network trained on-the-fly with

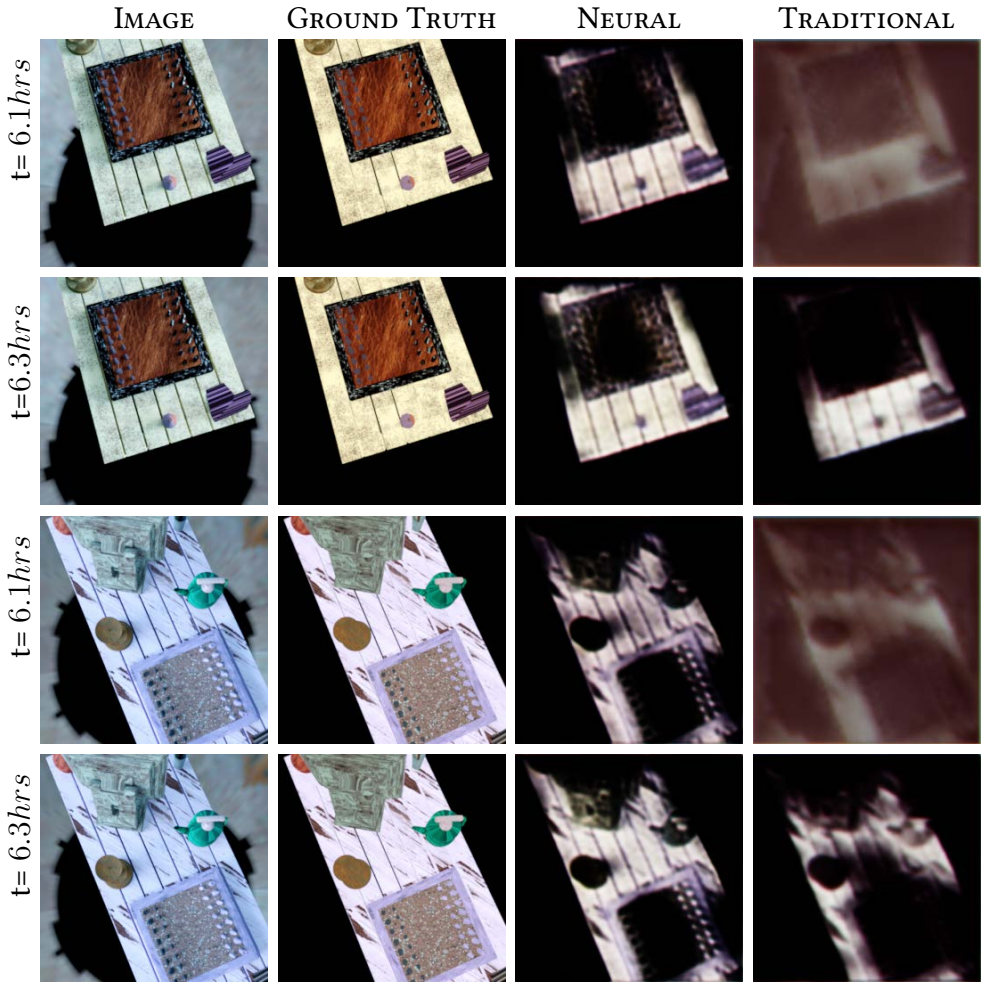


Figure 5.6: Results of training the intrinsic image decomposition network using Mitsuba 2 (Traditional) and on-the-fly neural (Neural) renderers. The input image is shown on the first column and the ground truth is shown on the second column. We observe at each timestamp the training has progressed significantly using neural renderer than traditional renderer as the prediction is visually much closer to ground truth for neural renderer than traditional renderer.

neural renderer converges faster.

5.5 Active Exploration Squared

On-the-fly data generation for training a network presents opportunities to generate samples which contribute most to the training. Diolatzis et al. [2022] demonstrated this for the task of learning the neural scene representation while generating data using the path tracer. They interleaved training with on-the-fly data generation to only generate

meaningful samples by exploring the data space using an *active exploration* strategy. We propose to do similar explorations of the data space to generate relevant data for training a network on a different task. In the process we use the neural renderer to generate data on-the-fly using the pre-trained model, which was also trained using active exploration, since it is faster than path tracing as we saw in Sec. 5.4. Thus, we use active exploration for learning the neural scene representation and generating data samples as well, which we term as *active exploration squared* strategy.

5.5.1 Methodology

In Sec. 5.3 we discussed Diolatzis et al. [2022] which defines a given scene configuration with an explicit scene representation vector that composes the data space of all possible variable scene configurations $\mathcal{D}$. They explored the space of scene configurations using Markov Chain Monte Carlo (MCMC) to guide sampling of the data space to maxmize contribution of the samples to training. They demonstrate that using MCMC explorations is beneficial especially for such high dimensional sample spaces such as the data space of variable scenes having pockets of importance.

The exploration defines a target function f and corresponding target distribution p such that the sampling proces produces samples which maximize training. Following previous work, the metric chosen to maximize training is the training loss and the norm of the total step while training. This divides the exploration into small steps and large steps. Small steps vary the scene configuration slightly exploring configurations in the neighborhood of the already explored space. Large steps changes the scene configurations significantly and helps the network overcome local minima.

As a first step, we will employ the same exploration strategy on a different task to accelerate the training. Our goal will be to define the target function for the specific task that we consider, for example intrinsic image decomposition. We can use the same metrics, i.e., the product of the training loss and norm of the total step where the training loss will be of the different task. We can then modify the target function depending on the requirement of the task as perceived. Using active exploration with on-the-fly data generation using neural rendering has potential to speed-up the traning process manyfolds by virtue of the speed of data generation and also because of generating meaningful samples which contribute to the training process.

5.6 Conclusion

We studied the feasibility of using neural renderers for synthetic data generation instead of traditional renderers such as a path tracer. We discussed recent neural rendering methods and implemented some additions to the recent neural renderer by Diolatzis et al. [2022] to use it for dataset generation. We compared the rendering rates of the neural renderer and the traditional path tracer Mitsuba 2 (Nimier-David et al. [2019]) and established the superiority of neural renderer in generating images faster. Comparing the two data generation methods on the task of intrinsic image decomposition, we saw the various trade-offs they offer in terms of computational budget allocation for training and data generation tasks. Taking one scenario as a case study, we presented initial results that shows using neural renderer with on-the-fly data generation to be beneficial over path tracers for training networks.

Using active exploration has been shown to be beneficial while generating data on-the-fly in case of learning neural scene representations (Diolatzis et al. [2022]). Our aim is to generalize the active exploration strategy to be used for training other tasks interleaved with on-the-fly data generation using neural rendering. We discussed how using variable scene representations in the form of explicit scene representation vector helps exploration of the data sample space. We plan to implement our active exploration squared strategy in the near future. In the process our goal is to develop a target function which is general enough for exploration of the data space for different learning tasks based on visual data.

Chapter 6

Conclusion

We started the thesis with the goal of rendering a captured scene faster and more efficiently while rendering with good quality preserving view-dependent appearance. We wanted to be cognizant of the surface appearance and use it to render and edit captured scene as well. We first proposed a new IBR algorithm, Hybrid-IBR, which renders at interactive frame rates with good quality preserving view-dependent effects based on a rough estimate of the underlying surface appearance. We moved on to explicitly estimate the surface appearance properties from multi-view captured images to produce 3D scene assets which can be used with a path tracer to render a scene with modified contents. We further presented neural rendering as a fast alternative to traditional rendering and studied its feasibility to serve as an on-the-fly dataset generator for training neural networks.

6.1 Thesis Summary

At the start of the thesis, many state-of-the-art IBR algorithms existed but most of them did not render at interactive frame rates. At the same time some algorithms required the textured mesh as a geometric proxy for intermediate tasks without actually rendering it. But the idea of falling back to the textured mesh in regions where the algorithms suffered, especially the background, was common. We envisioned an algorithm which uses the textured mesh in regions which are non-view dependent, or diffuse in nature, and more complex algorithms for other regions. Using the textured mesh came free of cost for significant parts of the rendered images which improved speed and made rendering more efficient.

We exploited the idea of using surface appearance for algorithm selection by estimating the *photometric uncertainty* of underlying surface. We further extended the idea of using more complex algorithms for poorly reconstructed regions by exclusively estimating the *geometric uncertainty*. Our prototype implementation of the Hybrid-IBR algorithm showed that a significant number of pixels in the final rendering can indeed be rendered

using cheaper algorithms. This helped us achieve our goal of rendering at interactive rates while maintaining good quality with view-dependent effects by identifying and treating non-diffuse regions explicitly. We were able to render a wide variety of scenes even on resource constrained devices such as a laptop at interactive frame rates. We believe that the idea can be further extended to render for more resource constrained devices such as mobile phones or virtual/mixed-reality headsets.

While we could use a rough estimate of the surface appearance to our advantage in developing a new IBR algorithm, it left more to be desired. Multi-view observations seemed to provide enough information to explicitly estimate surface material properties. Around the same time, many recent learning-based material estimation methods were being proposed. But most of the methods demonstrated material estimation on surface patches or planar objects at the time. We therefore wanted to come up with a learning-based method for the much harder case of scene-scale material estimation.

Scene-scale material estimation poses a much harder challenge than planar patches because of several reasons. Planar patches are free of occlusions and inter-reflections due to surrounding objects. These effects may change the appearance of the patch when captured. A scene by definition has multiple objects which occlude each other and can cast reflections and or shadows that makes one part of the observation different from the other part for the same material. The geometry of planar patches are simple and are implicitly assumed to be facing the camera. Scenes have much complex geometry and even while using state-of-the-art solutions to reconstruct the scene, significant errors still remain. Hence our algorithm needed to be robust to geometric errors. Furthermore, planar patches are often captured using co-located lighting with mobile flash to provide better visual cues to the network which is useful for specularity detection (Deschaintre et al. [2018, 2019]; Guo et al. [2020]; Asselin et al. [2020]). For scene-scale capture, mobile phone flash images do not provide the required cues and we had to rely on ambient scene illumination taking care to capture highlights from multiple views during capture.

Despite these challenges, scene-scale material estimation had not been explored in a casual capture context and it provides significant advantages. Having explicit surface material properties along with the 3D mesh of a captured scene is enough to render the scene with any edited scene configuration desired. We show a few use cases in rendering with new lighting configuration and inserting objects with complex physical properties.

We can even go further and edit the recovered material changing the scene appearance as desired. Going one step further they can help render the scene in real-time by pre-computing global illumination, as shown in recent works by Rodriguez et al. [2020a]; Rainer et al. [2022] or using realtime ray tracers such as Falcor (Kallweit et al. [2022]).

A key element for our material estimation task at scene-scale is the scene-scale synthetic dataset we rendered using Mitsuba renderer (Jakob [2010]). An extensive amount of time went into generating our dataset with significant variability. Despite our best efforts, it took us multiple iterations of the dataset to finally get the results we desired. Due to the number of iterations of the dataset that was needed, rendering costs using traditional rendering went into months and significantly slowed the advancement of our project. Due to the size of the dataset, storage, transfer of data, and I/O also took significant amount of time while providing satisfactory variability of the dataset. Thus after our second project we wanted a faster rendering alternative which gives significant variability in data as well.

Forward rendering with neural rendering holds promise to solve the data generation problem for training neural netowrks. While current neural rendering algorithms are at a nascent stage, with time they will evolve to support more complex rendering scenarios to generate images at-a-par with traditional rendering at a much faster rate. Nevertheless, we are studying the feasibility of using neural rendering in generating data on-the-fly. Recent methods (Diolatzis et al. [2022]) certainly have features such as speed and variability to be used as a dataset generator. Moreover the algorithms are flexible to meet the requirements of training as desired. We hope that future forward neural rendering algorithms keep this requirement in sight during their design process. Moreover a wider acceptance of neural renderers as dataset generators for training tasks will also need to be adopted by the machine learning community at large. We believe the community can only gain from it even beyond the domain of computer graphics.

6.2 Current Landscape

Our overall goal of the thesis was to render and edit a captured scenes faster and with more efficiency focusing on appearance. To do so we focussed our thesis on explicit scene assets in the form of MVS mesh for geometry and SVBRDF parameters for appearance based on a physically-based material model. We attempted to reduce geometric errors

for IBR using per-view mesh in Chapter 3 and we attempted to recover SVBRDF material properties to render an edited scene in Chapter 4. The Hybrid-IBR algorithm improved rendering rates but the quality is still limited due to the choice of algorithms we use for each pixel. In the three years of this thesis a number of works based on Neural Radiance Field (NeRF) (Mildenhall et al. [2020]; Martin-Brualla et al. [2021]) have been developed to render a captured scene which surpasses the quality we achieve. Nonetheless most of these works demonstrate their results on captures which focus on an object or a region of space instead of the full free-viewpoint navigation that we desire.

Even for the task of editing a scene's content, neural rendering is being exploited to learn a scene representation from captured images. The closest works are the NeRF-based works by Srinivasan et al. [2021]; Boss et al. [2020, 2021a,b] and the free-viewpoint indoor scene relighting work by Philip et al. [2021]. Among these the above problem of object-centric capture is still present with NeRF-based works. True free-viewpoint rendering with relighting is achieved only by Philip et al. [2021]. While the results are extremely encouraging, their method do not recover any intermediate scene properties such as material assets explicitly while using the MVS mesh as input. This observation begs a very interesting question: is there a need to recover human comprehensible classical scene assets if we can render and edit the scenes in neural latent space as demonstrated by Philip et al. [2021]?

One of the main motivations of trying to recover classical scene assets is their usage in CG and VFX industry. The graphics industry still relies on explicit scene assets generated by 3D artists because of its superior quality and artistic control. While the above mentioned NeRF-based works do try and query the NeRF to recover a mesh and surface materials in the form of SVBRDFs, the quality of the assets are not as per industrial standards. Efforts are ongoing in this direction to recover high quality mesh and appearance properties by both academia and the industry. Our work on material estimation was one of these attempts to use neural networks to explicitly learn physically-based material properties. While we showed that our recovered material maps can be used for editing scenes, we are still far from achieving industrial quality maps as generated by 3D artists.

6.3 Future Directions

In our attempt to make rendering faster, our Hybrid-IBR algorithm achieved faster rendering for IBR. For editing a scene we still rely on a traditional path tracer which is slow while we want to render the edited scene interactively. We can envision an immediate future work to this thesis with a new IBR algorithm to render an edited scene with global illumination using our recovered material maps potentially using NeRF-based geometry and input images. Our material maps can be edited based on user input to edit a scene interactively. This work can take inspiration from works which pre-compute global illumination using probes such as Rodriguez et al. [2020a]; McGuire et al. [2017].

We saw in Chapter 2 that representation and capture of materials have a long history in computer graphics and rendering. Recent methods based on neural networks have shown impressive results in tackling these two challenges. While we attempted capturing materials in Chapter 4, a future direction to work on may be the representation aspect. The problem is to represent high dimensional material data such as BTFs or SVBRDFs as a neural network, such that given the incoming and outgoing directions of the light source, we can get exact reflectance value thus compressing the data. Two recent works by Sztrajman et al. [2021]; Fan et al. [2022] have extended the idea to define a neural BRDF algebra and can now represent spatially varying materials with complex specular lobes. The next step could be to take the idea beyond the class of materials explored until now. While current works provide neural representations for opaque and/or shiny materials with layered coatings, we need to look into more complex materials categories such as BSSRDFs and BTDFs exhibiting transparent and/or translucent properties. The recent work by Deng et al. [2022] indicates this to be a promising avenue in the near future.

In Chapter 5 we focused on generating synthetic data using a neural renderer. Synthetic data with labels are required for learning-based tasks since it is hard to generate real annotated data. While the goal of using neural renderers is to be close to real data, if the domain gap is not closed, the learning algorithm suffers on real test cases. Multiple solutions to cross the synthetic to real domain gap such as Zhu et al. [2017]; Bi et al. [2019] have been proposed but no satisfactory solution exists yet. Studying domain transfer techniques to render more photorealistic images with characteristics of real images especially using neural renderers is also a very promising avenue for future work.

Appendix A

Hybrid Image-based Rendering

A.1 Computing the Harmonization mask

Akin to Agarwala et al. [2004] we define a cost function $C(l)$ for label l, where l is diffuse or view-dependent, i.e., $l \in \{\text{diff}, \text{spec}\}$, with a *unary cost* C_u over all pixels p and an *interaction cost* C_i over all pairs of pixels p and q in a 4-neighborhood

$$C(l) = \sum_{p} C_u(p, l_p) + \sum_{p,q} C_i(p, q, l_p, l_q). \tag{A.1}$$

This unary cost term C_u identifies view-dependent regions while the interaction cost term C_i tries to find good seams to minimize visible artifacts. We define

$$C_u(p, l_p) = \begin{cases} \infty & \text{if } l_p = \text{spec} \wedge p \notin R \\ \exp\left(\sigma_c + \lambda \Delta_c\right) & \text{if } l_p = \text{spec} \wedge p \in R \\ \exp\left(1 - (\sigma_c + \lambda \Delta_c)\right) & \text{if } l_p = \text{diff}. \end{cases}$$

The unary cost term considers the global color variance σ_c, the intensity difference between original and diffuse harmonized image Δ_c, and a spatial confidence region R. Ideally, we want the harmonized image to be similar to the original image. High color variance indicates specular regions (Lin et al. [2002]). Similarly, a high intensity difference between the original and diffuse harmonized images indicates highlight suppression. The parameter λ balances the relative importance of these two goals, and we set it to 2 in all our experiments. The spatial confidence margin penalizes specularities near the image edges to avoid re-introducing vignetting artifacts.

We define the interaction cost

$$C_i(p,q,l_p,l_q) = \begin{cases} \exp\left(\|c_p - c_q\|^2\right) & \text{if } l_p = l_q \\ \exp\left(1 - \|c_p - c_q\|^2\right) & \text{if } l_p \neq l_q, \end{cases}$$

where c denote the colors of the original image. If the color difference of neighboring pixels is high we penalize assigning same labels to both pixels since the pixels may constitute a true edge that we want to preserve.

A.2 Comparison code

We used published implementations for previous work. Specifically, for Hedman et al. [2018]: `https://sibr.gitlabpages.inria.fr`, for Mildenhall et al. [2020]: `https://github.com/bmild/nerf`, for Riegler and Koltun [2020] `https://github.com/intel-isl/FreeViewSynthesis`, and for the perceptual error metric E-LPIPS Kettunen et al. [2019]: `https://github.com/mkettune/elpips`.

For NeRF, we had to select a subset of cameras, since if we used all of them the results were unusable. We tried different combinations of cameras, and kept the one with the best visual result.

A.3 Additional Results and Preprocessing Statistics

A.3.1 Runtime Comparisons

We compare the total runtime of different algorithms on the 2 machines described in the main paper for a given dataset in Table A.1.

Table A.1: Comparison of total runtime of different algorithms over a sequence of 100 frames on 2 different machines with 2 datasets for a rendering resolution of 1280x720.

IBR Algorithms	Desktop		Laptop	
	Hugo	Dr Johnson	Hugo	Dr Johnson
ULR	0.06 ms	7.32 ms	0.01 ms	14.45 ms
InsideOut	8.37 ms	12.93 ms	10.63 ms	18.24 ms
Deep Blending	64 ms	68.78 ms	79.89 ms	98.88 ms
Ours	21.11 ms	31.59 ms	27.12 ms	52.03 ms

A.3.2 Test Image Set

We show all images used for testing with ground truth from the Ponche dataset in Figure A.1 and from the Synthetic Attic dataset in Figure A.2.

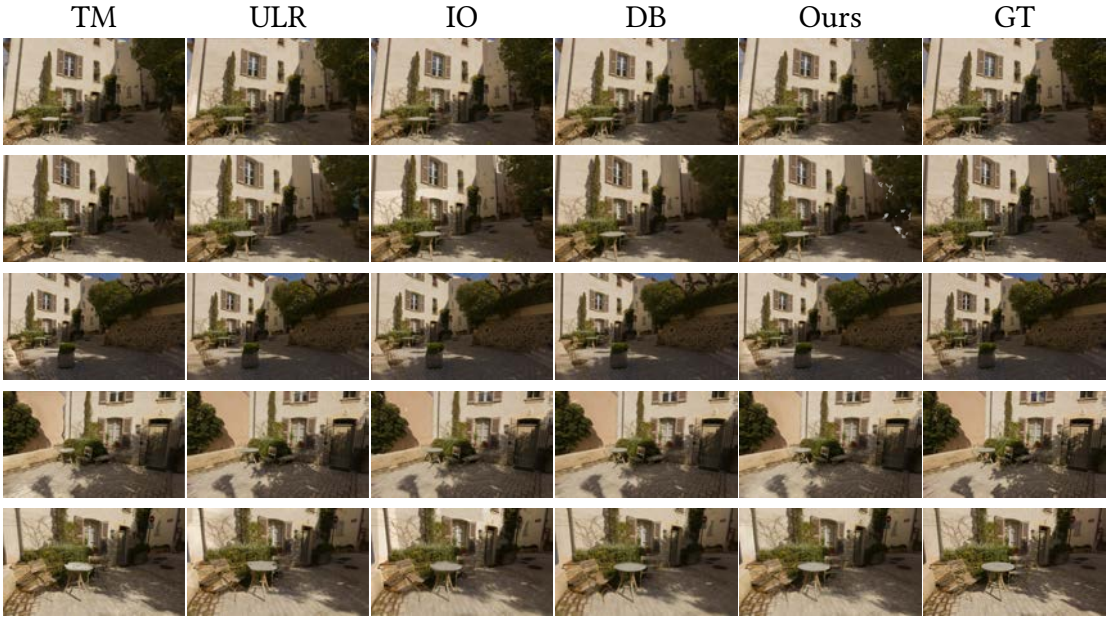


Figure A.1: Held-out test images with their ground truth input image from Ponche. TM: Textured Mesh (Reality [2018]); ULR: Unstructured Lumigraph (Buehler et al. [2001]); IO: Inside Out (Hedman et al. [2016]); DB: Deep Blending (Hedman et al. [2018]); GT: Ground Truth.

A.3.3 Pre-processing Time

We also report the pre-processing time with a breakdown of each component in Table A.2. The pre-processing time varies based on number of images and resolution of original images. Hence, we provide the numbers for 3 different datasets to give a better insight into pre-processing time.

A.3.4 Dataset Statistics

We provide additional information on our datasets reporting total number of images, input image resolutions, and total processing time in Table A.3. All input images are Low Dynamic Range (LDR) captured using commercial hand-held devices (typical SLR Cameras) in an unconstrained or casual acquisition process. We resize the input images

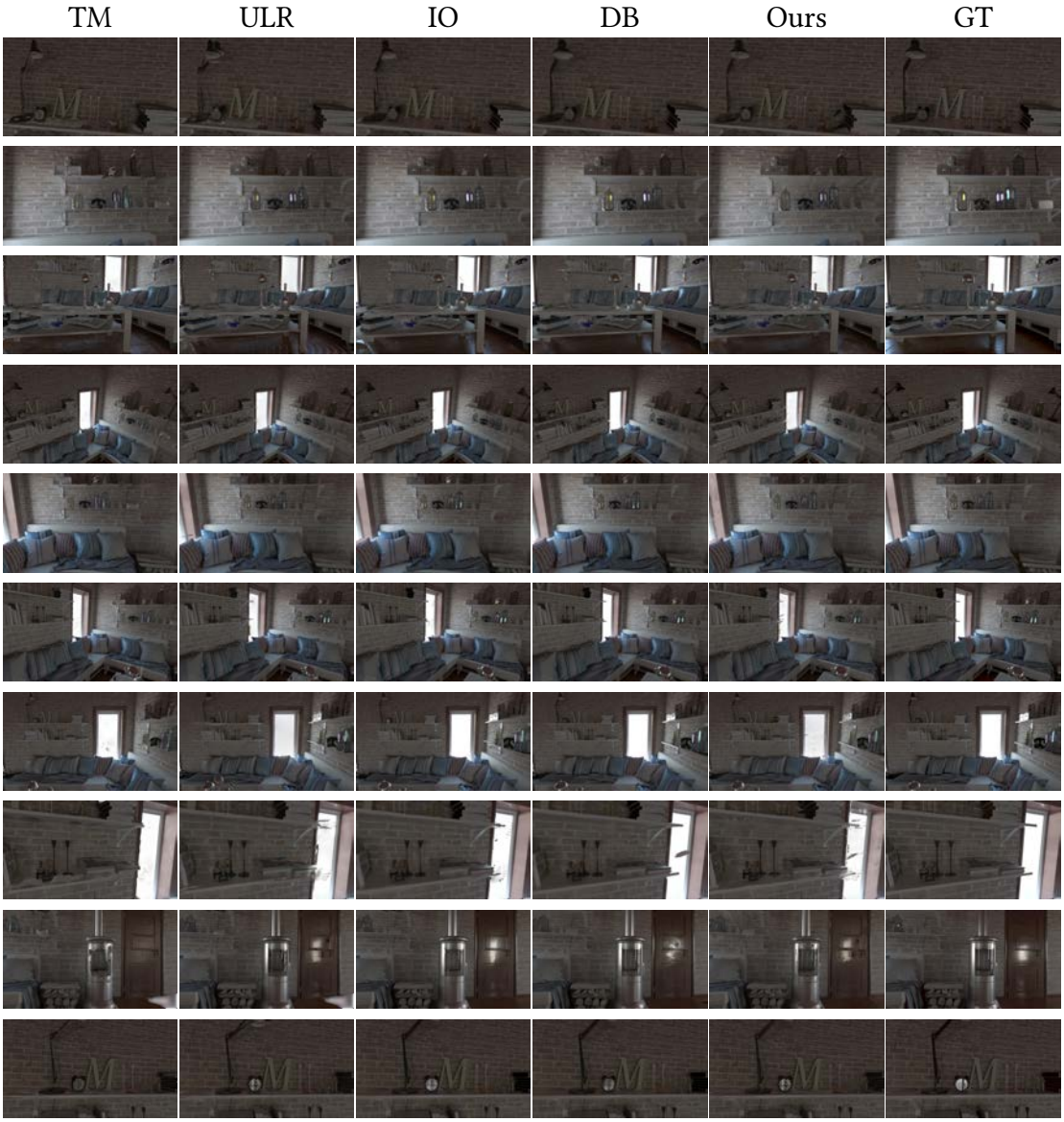


Figure A.2: Novel view test images with their path traced ground truth from Synthetic Attic. TM: Textured Mesh (Reality [2018]); ULR: Unstructured Lumigraph (Buehler et al. [2001]); IO: Inside Out (Hedman et al. [2016]); DB: Deep Blending (Hedman et al. [2018]); GT: Ground Truth.

to 1920px in the dominant resolution maintaining the aspect ratio to fit all data on the GPU VRAM. We do not support datasets which scale above the GPU VRAM limit but a future extension by using streaming architectures is an interesting follow-up to our method.

Table A.2: Pre-processing time for Hugo (24 images, 3216X2136), Creepy Attic (246 images, 1228X816), and Library (222 images, 3552X2000) on a system with Intel Xeon Gold 5218 2.30GHz Processor and Quadro RTX 5000 GPU.

Component	Hugo	Creepy Attic	Library
Harmonization Step 1	4m	6m	38m
Harmonization Step 2	9m	14m	1h 58m
Spec Mask Texturing	26s	1m 13s	1m 21s
Tiling and Storing	16s	1m 33s	1m 37s
Total	14m	26m	2h 40m

Table A.3: Statistics for all datasets presented in the paper.

Dataset	#Images	Image Res.	Preprocess Time (Total)
Hugo	24	$3216X2136$	14 min
Ponche	50	$2016X1344$	19 min
Library	222	$3552X2000$	2h 40 min
Playroom	226	$1264X832$	25 min
Creepy Attic	246	$1228X816$	26 min
Dr Johnson	264	$1296X864$	31 min
Syn. Attic	283	$1920X1080$	57 min
Train	301	$1920X1080$	2h 8 min
Salon	344	$3000X2000$	3h 45 min

Appendix B

Material estimation from indoor captures

B.1 Network Architecture Details

We provide network details for the diffuse track and specular track in Tables B.1 and B.2 respectively for an image of resolution $640 \times 384 \times 3$. As described in Section 4.2 in the main paper, the number of input channel is defined by the multi-view statistics being fed to each network. In the decoder from layer 4 onwards, a nearest-neighbor upsampling is followed by concatenation of encoder features, and two convolutions. For activations, each convolution/de-convolution layer is followed by a leaky ReLU with a weight 0.2 for the negative part.

Input	Layer	Output
In: $640 \times 384 \times 9$	4×4 Conv, 64, stride 2	**1**: $320 \times 192 \times 64$
0: 9	FC + SeLU	**1**: 128
1: $320 \times 192 \times 64$	4×4 Conv, 128, stride 2	**2**: $160 \times 96 \times 128$
1: 256	FC + SeLU	**2**: 256
2: $160 \times 96 \times 128$	4×4 Conv, 256, stride 2	**3**: $80 \times 48 \times 256$
2: 512	FC + SeLU	**3**: 512
3: $80 \times 48 \times 256$	4×4 Conv, 512, stride 2	**4**: $40 \times 24 \times 512$
3: 1024	FC + SeLU	**4**: 512
4: $40 \times 24 \times 512$	4×4 DeConv, 256, stride 1	**5**: $80 \times 48 \times 256$
4: 768	FC + SeLU	**5**: 256
5: $80 \times 48 \times 512$	4×4 DeConv, 128, stride 1	**6**: $160 \times 96 \times 128$
5: 384	FC + SeLU	**6**: 180
6: $160 \times 96 \times 256$	4×4 DeConv, 64, stride 1	**7**: $320 \times 192 \times 64$
6: 192	FC + SeLU	**7**: 64
7: $320 \times 192 \times 128$	4×4 DeConv, 3, stride 1	**Out**: $640 \times 384 \times 3$
7: 67	FC + SeLU	**8**: 3

Table B.1: Details of the network architecture for diffuse track.

Input	Layer	Output
In: $640 \times 384 \times 18$	4×4 Conv, 64, stride 2	**1**: $320 \times 192 \times 64$
0: 9	FC + SeLU	**1**: 128
1: $320 \times 192 \times 64$	4×4 Conv, 128, stride 2	**2**: $160 \times 96 \times 128$
1: 256	FC + SeLU	**2**: 256
2: $160 \times 96 \times 128$	4×4 Conv, 256, stride 2	**3**: $80 \times 48 \times 256$
2: 512	FC + SeLU	**3**: 512
3: $80 \times 48 \times 256$	4×4 Conv, 512, stride 2	**4**: $40 \times 24 \times 512$
3: 1024	FC + SeLU	**4**: 512
4: $40 \times 24 \times 512$	4×4 DeConv, 256, stride 1	**5**: $80 \times 48 \times 256$
4: 768	FC + SeLU	**5**: 256
5: $80 \times 48 \times 512$	4×4 DeConv, 128, stride 1	**6**: $160 \times 96 \times 128$
5: 384	FC + SeLU	**6**: 180
6: $160 \times 96 \times 256$	4×4 DeConv, 64, stride 1	**7**: $320 \times 192 \times 64$
6: 192	FC + SeLU	**7**: 64
7: $320 \times 192 \times 128$	4×4 DeConv, 4, stride 1	**Out**: $640 \times 384 \times 4$
7: 67	FC + SeLU	**8**: 4

Table B.2: Details of the network architecture for specular track.

B.2 Additional Results

We provide additional results for our experiments.

B.2.1 Ablation

Figure B.1 shows the effects of increasing multi-view information for the Synthetic Veach Ajar scene. This figure is a extension of Fig. 12 in the main paper.

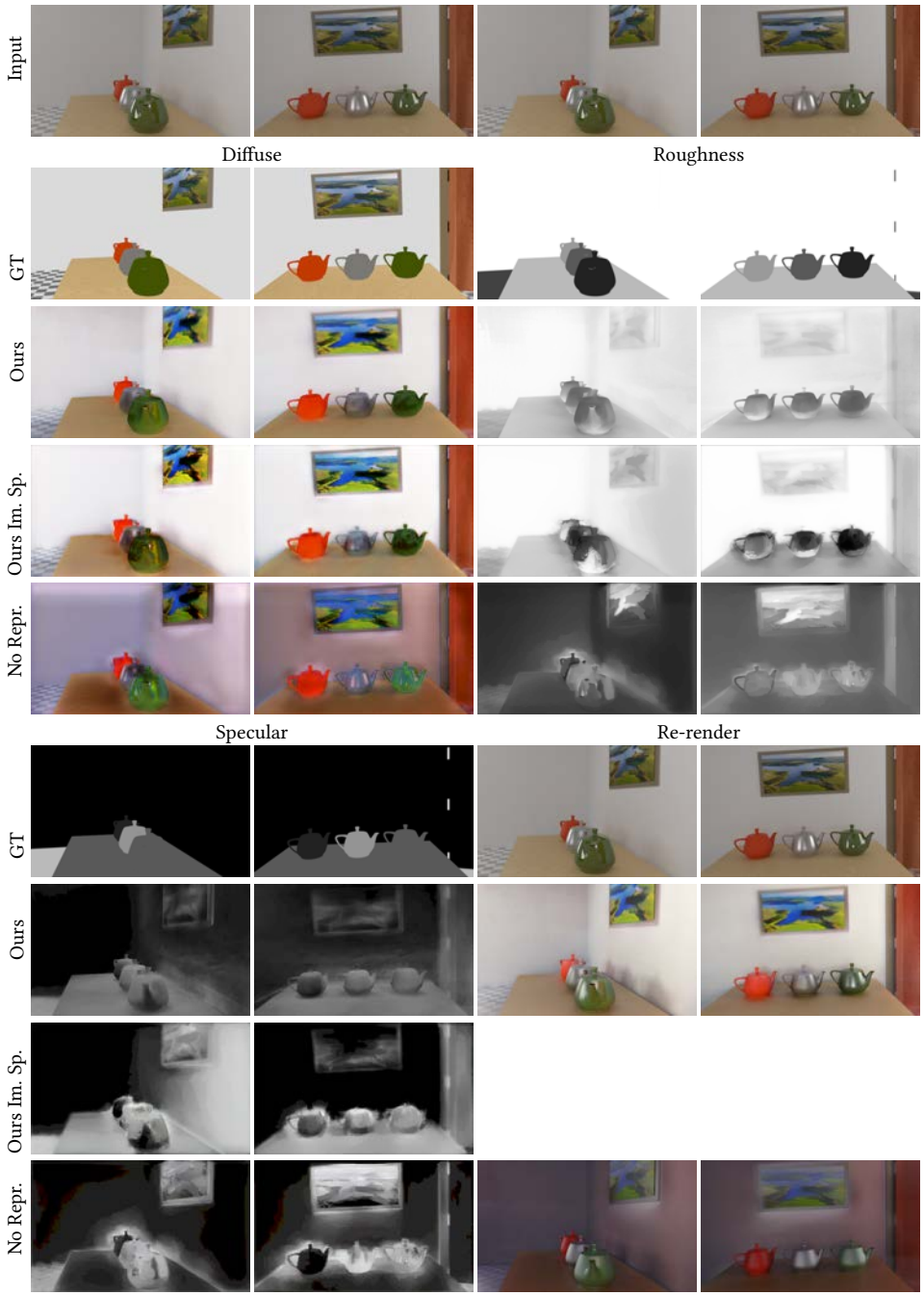


Figure B.1: Example images from Synthetic Veach Ajar showing the effect of increasing multi-view information on results. Note how the quality of the maps is significantly improved by using the reprojected statistics observed in our image space predicted maps as compared to no reprojection, i.e. using only a single image. Furthermore, gathering the image space maps in texture space helps improve the consistency of the maps across views and thus improves re-rendering by assigning same material in local regions (esp. in roughness and specular maps). As a result, we get better re-rendering as the re-rendered images are closer to ground truth.

Bibliography

Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 294–302, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 9781450378239. doi: 10.1145/1186562.1015718. URL https://doi.org/10.1145/1186562.1015718. 35, 44, 45, 113

Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Practical svbrdf capture in the frequency domain. *ACM Trans. Graph.*, 32(4), July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461978. URL https://doi.org/10.1145/2461912.2461978. 24, 25

Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Trans. Graph.*, 34(4), July 2015. ISSN 0730-0301. doi: 10.1145/2766967. URL https://doi.org/10.1145/2766967. 25, 28

Anthony A Apodaca and MW Mantle. Renderman: Pursuing the future of graphics. *IEEE Computer Graphics and Applications*, 10(4):44–49, 1990. 12

Anthony A Apodaca, Larry Gritz, and Ronen Barzel. *Advanced RenderMan: Creating CGI for motion pictures*. Morgan Kaufmann, 2000. 12

Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, 1968. 1, 9

James Richard Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, USA, 1995. AAI9619140. 2, 11, 18

Michael Ashikhmin and Peter Shirley. An anisotropic phong brdf model. *J. Graph. Tools*, 5(2):25–32, feb 2000. ISSN 1086-7651. doi: 10.1080/10867651.2000.10487522. URL https://doi.org/10.1080/10867651.2000.10487522. 22, 23

L. Asselin, D. Laurendeau, and J. Lalonde. Deep svbrdf estimation on real materials. In *2020 International Conference on 3D Vision (3DV)*, pages 1157–1166, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society. doi: 10.1109/3DV50981.2020.00126. URL https://doi.ieeecomputersociety.org/10.1109/3DV50981.2020.00126. 28, 63, 66, 108

Dejan Azinovic, Tzu-Mao Li, Anton Kaplanyan, and Matthias Niessner. Inverse path tracing for joint material and lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 26, 62

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. NeRF-Tex: Neural Reflectance Field Textures. In Adrien Bousseau and Morgan McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021. ISBN 978-3-03868-157-1. doi: 10.2312/sr.20211285. 31

Réjean Baribeau, William S. Neil, and Éric Côté. Development of a robot-based gonioreflectometer for spectral brdf measurement. *Journal of Modern Optics*, 56(13): 1497–1503, 2009. doi: 10.1080/09500340903045702. URL `https://doi.org/10.1080/09500340903045702`. 25

Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, 2015. doi: 10.1109/TPAMI.2014.2377712. 19

Jonathan T. Barron and Ben Poole. The fast bilateral solver. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 617–632, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9. 68, 69

Jean-Philippe Bauchet and Florent Lafarge. Kinetic Shape Reconstruction. *ACM Trans. on Graphics*, 39(5), 2020. 84

Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Trans. Graph.*, 39(6), nov 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417827. URL `https://doi.org/10.1145/3414685.3417827`. 30

Sai Bi, Kalyan Sunkavalli, Federico Perazzi, Eli Shechtman, Vladimir Kim, and Ravi Ramamoorthi. Deep cg2real: Synthetic-to-real translation via image disentanglement. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2730–2739, 2019. doi: 10.1109/ICCV.2019.00282. 111

Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, and Ravi Ramamoorthi. Deep 3d capture: Geometry and reflectance from sparse multi-view images. In *2020 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 5959–5968, 2020. doi: 10.1109/CVPR42600.2020.00600. 28, 62

James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, page 192–198, New York, NY, USA, 1977. Association for Computing Machinery. ISBN 9781450373555. doi: 10.1145/563858.563893. URL https://doi.org/10.1145/563858.563893. 22, 23, 25

Sebastien Bonopera, Peter Hedman, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. sibr: A system for image based rendering, 2020. URL https://sibr.gitlabpages.inria.fr/. 14, 44, 50

Mark Boss, Varun Jampani, Kihwan Kim, Hendrik P.A. Lensch, and Jan Kautz. Two-shot spatially-varying brdf and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 28, 63, 69, 110

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12684–12694, October 2021a. 30, 63, 110

Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021b. 30, 63, 110

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 425–432, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113374X. doi: 10.1145/383259.383309. URL https://doi.org/10.1145/383259.383309. 6, 13, 15, 18, 34, 35, 38, 42, 43, 47, 52, 56, 115, 116

Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *ACM SIGGRAPH*, volume 2012, pages 1–7. vol. 2012, 2012. 12

Chaos Group. V-ray. https://www.chaosgroup.com/3d-rendering-software, 2017. 12

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):30, 2013. 14, 15, 18, 34, 38

Shenchang Eric Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 29–38, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917014. doi: 10.1145/218380.218395. URL https://doi.org/10.1145/218380.218395. 2, 12, 37

Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 279–288, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897916018. doi: 10.1145/166117.166153. URL https://doi.org/10.1145/166117.166153. 12, 37

Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, et al. Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG)*, 37(3):1–21, 2018. 12

R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, jan 1982. ISSN 0730-0301. doi: 10.1145/357290.357293. URL https://doi.org/10.1145/357290.357293. 22, 23, 24, 65

Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, January 1999. ISSN 0730-0301. doi: 10.1145/300776.300778. URL https://doi.org/10.1145/300776.300778. 22, 25

Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, pages 105–116. Springer Vienna, Vienna, 1998. ISBN 978-3-7091-6453-2. 3, 13, 18, 38

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 11–20, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237191. URL https://doi.org/10.1145/237170.237191. 3, 13, 38

Xi Deng, Fujun Luan, Bruce Walter, Kavita Bala, and Steve Marschner. Reconstructing translucent objects using differentiable rendering. *ACM Trans. Graph.*, July 2022. 111

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201378. URL https://doi.org/10.1145/3197517.3201378. 5, 27, 60, 61, 63, 65, 66, 68, 69, 70, 71, 108

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Flexible SVBRDF Capture with a Multi-Image Deep Network. *Computer Graphics Forum*, 2019. ISSN 1467-8659. doi: 10.1111/cgf.13765. 5, 28, 63, 65, 66, 68, 108

Valentin Deschaintre, Yiming Lin, and Abhijeet Ghosh. Deep polarization imaging for 3d shape and svbrdf acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15567–15576, June 2021. 28

Stavros Diolatzis, Julien Philip, and George Drettakis. Active exploration for neural global illumination of variable scenes. *ACM Trans. Graph.*, 41(5), may 2022. ISSN 0730-0301. doi: 10.1145/3522735. URL https://doi.org/10.1145/3522735. viii, 18, 30, 31, 91, 92, 93, 96, 97, 98, 100, 104, 105, 106, 109

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Computer Graphics Forum*, 27(2):409–418, 2008. doi: https://doi.org/10.1111/j.1467-8659.2008.01138.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01138.x. 14, 34, 38

S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum,

Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. doi: 10.1126/science.aar6170. URL `https://www.science.org/doi/abs/10.1126/science.aar6170`. 29, 30, 96, 97

Jiahui Fan, Beibei Wang, Miloš Hašan, Jian Yang, and Ling-Qi Yan. Neural layered brdfs. In *Proceedings of SIGGRAPH 2022*, 2022. 111

Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, and Brian Green. Path tracing in production. In *ACM SIGGRAPH 2017 Courses*, pages 1–39. 2017. 60

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 17, 39

Basile Fraboni, Jean-Claude Iehl, Vincent Nivoliers, and Guillaume Bouchard. Adaptive Multi-view Path Tracing. In *Eurographics Symposium on Rendering (EGSR)*, 2019. 87

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323042. URL `https://doi.org/10.1145/3306346.3323042`. 27, 63

Duan Gao, Guojun Chen, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deferred neural lighting: Free-viewpoint relighting from unstructured photographs. *ACM Trans. Graph.*, 39(6), November 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417767. URL `https://doi.org/10.1145/3414685.3417767`. 31

Elena Garces, Carlos Rodriguez-Pardo, Dan Casas, and Jorge Lopez-Moreno. A survey on intrinsic images: Delving deep into lambert and beyond. *International Journal in Computer Vision 130, 836-868*, 2022. 24

Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *ACM Trans. Graph.*, 36(6), November 2017. ISSN 0730-0301.

doi: 10.1145/3130800.3130891. URL https://doi.org/10.1145/3130800.3130891. 77

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, et al. Arnold: A brute-force production path tracer. *ACM Transactions on Graphics (TOG)*, 37 (3):1–12, 2018. 12

Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Estimating specular roughness and anisotropy from second order spherical gradient illumination. In *Proceedings of the Twentieth Eurographics Conference on Rendering*, EGSR'09, page 1161–1170, Goslar, DEU, 2009. Eurographics Association. doi: 10.1111/j.1467-8659.2009.01493.x. URL https://doi.org/10.1111/j.1467-8659.2009.01493.x. 25

Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Circularly polarized spherical illumination reflectometry. *ACM Trans. Graph.*, 29 (6), December 2010. ISSN 0730-0301. doi: 10.1145/1882261.1866163. URL https://doi.org/10.1145/1882261.1866163. 24, 25

Andrew Glassner. Deep learning: A crash course. In *ACM SIGGRAPH 2019 Courses*, SIGGRAPH '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450363075. doi: 10.1145/3305366.3328026. URL https://doi.org/10.1145/3305366.3328026. 91

P. Goel, L. Cohen, J. Guesman, V. Thamizharasan, J. Tompkin, and D. Ritchie. Shape from tracing: Towards reconstructing 3d object geometry and svbrdf material from images via differentiable path tracing. In *2020 International Conference on 3D Vision (3DV)*, pages 1186–1195, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society. doi: 10.1109/3DV50981.2020.00129. URL https://doi.ieeecomputersociety.org/10.1109/3DV50981.2020.00129. 27, 62

M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. doi: 10.1109/ICCV.2007.4408933. 33

Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. Ambient point clouds for view interpolation. In

ACM SIGGRAPH 2010 Papers, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450302104. doi: 10.1145/1833349.1778832. URL https://doi.org/10.1145/1833349.1778832. 14, 38

Daniel B Goldman. Vignette and exposure calibration and compensation. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2276–2288, 2010. 37

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 43–54, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237200. URL https://doi.org/10.1145/237170.237200. 13, 38

Jonathan Granskog, Fabrice Rousselle, Marios Papas, and Jan Novák. Compositional neural scene representations for shading inference. *ACM Trans. Graph.*, 39(4), jul 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392475. URL https://doi.org/10.1145/3386569.3392475. 29, 30, 96, 97

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. 2022. 95

D. Guarnera, G.C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. Brdf representation and acquisition. *Computer Graphics Forum*, 35(2):625–650, 2016. doi: https://doi.org/10.1111/cgf.12867. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12867. 21, 24

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. Highlight-aware two-stream network for single-image svbrdf acquisition. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459854. URL https://doi.org/10.1145/3450626.3459854. 27, 60, 63, 65

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. Materialgan: Reflectance capture using a generative svbrdf model. *ACM Trans. Graph.*, 39 (6), November 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417779. URL `https://doi.org/10.1145/3414685.3417779`. 28, 29, 63, 108

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)*, 30(4):1–10, 2011. 37

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Optimizing color consistency in photo collections. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 34, 37

Bjoern Haefner, Simon Green, Alan Oursland, Daniel Andersen, Michael Goesele, Daniel Cremers, Richard Newcombe, and Thomas Whelan. Recovering real-world reflectance properties and shading from hdr imagery. In *2021 International Conference on 3D Vision (3DV)*, pages 1075–1084, 2021. doi: 10.1109/3DV53792.2021.00115. 26, 62, 77, 79

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):231, 2016. 6, 14, 15, 16, 18, 38, 39, 41, 46, 47, 51, 52, 55, 56, 115, 116

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275084. URL `https://doi.org/10.1145/3272127.3275084`. 3, 6, 15, 16, 18, 34, 35, 36, 38, 39, 40, 41, 42, 43, 46, 48, 51, 52, 53, 55, 56, 114, 115, 116

Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. Generative modelling of brdf textures from flash images. *arXiv preprint arXiv:2102.11861*, 2021. 27, 63

Silja Holopainen, Farshid Manoocheri, Erkki Ikonen, Kai-Olaf Hauer, and Andreas Höpe. Comparison measurements of 0:45 radiance factor and goniometrically determined diffuse reflectance. *Appl. Opt.*, 48(15):2946–2956, May 2009. doi: 10.1364/AO.48.002946. URL `https://opg.optica.org/ao/abstract.cfm?URI=ao-48-15-2946`. 25

Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 3dlite: towards commodity 3d scanning for content creation. *ACM Trans. Graph.*, 36(6):203–1, 2017. 34, 37, 45, 53

Wenzel Jakob. Mitsuba renderer, 2010. 12, 18, 24, 64, 71, 88, 91, 94, 109

M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*, pages 3121–3128, 2011. doi: 10.1109/CVPR.2011.5995693. 37, 38, 51

James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery. ISBN 0897911962. doi: 10.1145/15922.15902. URL https://doi.org/10.1145/15922.15902. 1, 10

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. The Falcor rendering framework, 8 2022. URL https://github.com/NVIDIAGameWorks/Falcor. https://github.com/NVIDIAGameWorks/Falcor. 12, 18, 109

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3D shape segmentation with projective convolutional networks. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. 88

Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308076. doi: 10.1145/2024156.2024191. URL https://doi.org/10.1145/2024156.2024191. 19, 77

Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*, 33(3), June 2014. ISSN 0730-0301. doi: 10.1145/2602146. URL https://doi.org/10.1145/2602146. 19, 77

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 24

Douglas Scott Kay and Donald Greenberg. Transparency for computer synthesized images. *SIGGRAPH Comput. Graph.*, 13(2):158–164, aug 1979. ISSN 0097-8930. doi: 10.1145/965103.807438. URL https://doi.org/10.1145/965103.807438. 9

Alexander Keller, Jaroslav Křivánek, Jan Novák, Anton Kaplanyan, and Marco Salvi. Machine learning and rendering. In *ACM SIGGRAPH 2018 Courses*, SIGGRAPH '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450358095. doi: 10.1145/3214834.3214841. URL https://doi.org/10.1145/3214834.3214841. 91

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. E-lpips: Robust perceptual image similarity via random transformation ensembles, 2019. 55, 70, 114

Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Trans. Graph.*, 33(4), jul 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601209. URL https://doi.org/10.1145/2601097.2601209. 3, 19

Seon Joo Kim and Marc Pollefeys. Robust radiometric calibration and vignetting correction. *IEEE transactions on pattern analysis and machine intelligence*, 30(4):562–576, 2008. 37

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 74

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073599. URL https://doi.org/10.1145/3072959.3073599. 51, 53

Vladlen Koltun. Towards photorealism. 42nd German Conference on Pattern Recognition (DAGM GCPR 2020), the 25th International Symposium on Vision, Modeling and

Visualization (VMV 2020) and the 10th Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM 2020), 2020. URL https://youtu.be/Rd0nBO6--bM. 39

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 40(4):29–43, 2021. doi: https://doi.org/10.1111/cgf.14339. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14339. 17

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)*, 32(6):199, 2013. 4, 15, 18, 38

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG)*, 22 (3):277–286, 2003. ISSN 0730-0301. 37, 45

Sébastien Lachambre, Sébastien Lagarde, and Jover Cyril. Unity photogrammetry workflow. https://unity3d.com/files/solutions/photogrammetry/Unity-Photogrammetry-Workflow_2017-07_v2.pdf, 2017. 60, 65

Eric Lafortune. Mathematical models and monte carlo algorithms for physically based rendering. *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven*, 20:74–79, 1996. 2

Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993. 11

V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007. doi: 10.1109/CVPR.2007.383078. 37

Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 31–42, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237199. URL https://doi.org/10.1145/237170.237199. 3, 12, 38

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073641. URL https://doi.org/10.1145/3072959.3073641. 27, 65

Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 20, 92, 94, 101

Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. Materials for masses: SVBRDF acquisition with a single mobile phone image. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, volume 11207 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 2018a. doi: 10.1007/978-3-030-01219-9_5. URL https://doi.org/10.1007/978-3-030-01219-9_5. 27, 65

Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Trans. Graph.*, 37(6), December 2018b. ISSN 0730-0301. doi: 10.1145/3272127.3275055. URL https://doi.org/10.1145/3272127.3275055. 28, 60, 61, 63, 69

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 28, 60, 61, 63, 68, 69, 79, 92, 95

Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, Sai Bi, Hong-Xing Yu, Zexiang Xu, Kalyan Sunkavalli, Milos Hasan, Ravi Ramamoorthi, and Manmohan Chandraker. Openrooms: An open framework for photorealistic indoor scene datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7190–7199, June 2021. 64, 95

Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Milos Hasan, Kalyan Sunkavalli, and Manmohan Chandraker. Photoscene: Photorealistic material and lighting transfer for indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 28, 63

Stephen Lin, Yuanzhen Li, Sing Bing Kang, Xin Tong, and Heung-Yeung Shum. Diffuse-specular separation and depth recovery from image sequences. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision — ECCV 2002*, pages 210–224, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-47977-2. 113

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323020. URL `https://doi.org/10.1145/3306346.3323020`. 17, 39

A. Loza, L. Mihaylova, N. Canagarajah, and D. Bull. Structural similarity-based object tracking in video sequences. In *2006 9th International Conference on Information Fusion*, pages 1–6, 2006. doi: 10.1109/ICIF.2006.301574. 55, 79

Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. *Computer Graphics Forum*, 40(4):101–113, 2021. doi: https://doi.org/10.1111/cgf.14344. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14344`. 27, 62

Xiaohe Ma, Kaizhang Kang, Ruisheng Zhu, Hongzhi Wu, and Kun Zhou. Free-form scanning of non-planar appearance with neural trace photography. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459679. URL `https://doi.org/10.1145/3450626.3459679`. 27

Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. Image-based brdf measurement including human skin. In *Proceedings of the 10th Eurographics Conference on Rendering*, EGWR'99, page 131–144, Goslar, DEU, 1999. Eurographics Association. ISBN 321183382X. 25

Stephen Robert Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, USA, 1998. AAI9839924. 3, 25

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections, 2021. 18, 39, 110

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, jul 2003a. ISSN 0730-0301. doi: 10.1145/882262.882343. URL https://doi.org/10.1145/882262.882343. 22, 25

Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic brdf measurement. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, page 241–247, Goslar, DEU, 2003b. Eurographics Association. ISBN 3905673037. 22, 25

Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348867. doi: 10.1145/3023368.3023378. URL https://doi.org/10.1145/3023368.3023378. 111

Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 39–46, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917014. doi: 10.1145/218380.218398. URL https://doi.org/10.1145/218380.218398. 12, 37

Abhimitra Meka, Maxim Maximov, Michael Zollhöfer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 20, 101

Abhimitra Meka, Rohit Pandey, Christian Haene, Sergio Orts-Escolano, Peter Barnum, Philip Davidson, Daniel Erickson, Yinda Zhang, Jonathan Taylor, Sofien Bouaziz, Chloe Legendre, Wan-Chun Ma, Ryan Overbeck, Thabo Beeler, Paul Debevec, Shahram Izadi, Christian Theobalt, Christoph Rhemann, and Sean Fanello. Deep relightable textures - volumetric performance capture with neural rendering. volume 39, December 2020. doi: 10.1145/3414685.3417814. URL http://gvv.mpi-inf.mpg.de/projects/DeepRelightableTextures/. 31

Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 17, 39

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322980. URL `https://doi.org/10.1145/3306346.3322980`. 17, 39, 43

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 405–421, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8. 17, 18, 29, 30, 39, 51, 52, 110, 114

Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. Openmvg: Open multiple view geometry. In Bertrand Kerautret, Miguel Colom, and Pascal Monasse, editors, *Reproducible Research in Pattern Recognition*, pages 60–74, Cham, 2017. Springer International Publishing. 37

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459812. URL `https://doi.org/10.1145/3450626.3459812`. 30

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), jul 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530127. URL `https://doi.org/10.1145/3528223.3530127`. 18

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. *arXiv:2111.12503*, 2021. 30, 64

O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel. Deep shading: Convolutional neural networks for screen space shading. *Comput. Graph. Forum*, 36(4): 65–78, jul 2017. ISSN 0167-7055. doi: 10.1111/cgf.13225. URL `https://doi.org/10.1111/cgf.13225`. 30

Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H. Kim. Practical svbrdf acquisition of 3d objects with unstructured flash photography. *ACM Trans. Graph.*, 37(6), December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275017. URL `https://doi.org/10.1145/3272127.3275017`. 27, 62

Fred E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4(7):767–775, Jul 1965. doi: 10.1364/AO.4.000767. URL `http://opg.optica.org/ao/abstract.cfm?URI=ao-4-7-767`. 20, 25

Fred Edwin Nicodemus, RICHMOND JC, et al. Geometrical considerations and nomenclature for reflectance. 1977. 20

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6): 1–17, 2019. 12, 18, 93, 98, 100, 106

Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In Adrien Bousseau and Morgan McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021. ISBN 978-3-03868-157-1. doi: 10.2312/sr.20211292. 26, 60, 62, 77, 79

Gael Obein, Robert Bousquet, and Maria Nadal. New nist reference goniospectrometer. Optics and Photonics Symposium, 2005-08-01 2005. doi: https://doi.org/10.1117/12.621516. 25

Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 239–246, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916670. doi: 10.1145/192161.192213. URL `https://doi.org/10.1145/192161.192213`. 22, 24

R. Ortiz-Cayon, A. Djelouah, and G. Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *2015 International Conference on 3D Vision*, pages 469–477, 2015. doi: 10.1109/3DV.2015.59. 38

Romain Pacanowski, Oliver Salazar Celis, Christophe Schlick, Xavier Granier, Pierre Poulin, and Annie Cuyt. Rational brdf. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1824–1835, 2012. doi: 10.1109/TVCG.2012.73. 22

Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: a general purpose ray tracing engine. *Acm transactions on Graphics (TOG)*, 29(4):1–13, 2010. 71

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 73

Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):235, 2017. 3, 15

Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, page 313–318. Association for Computing Machinery, New York, NY, USA, 2003. ISBN 1581137095. doi: 10.1145/1201775.882269. URL `https://doi.org/10.1145/1201775.882269`. 45

Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 12

Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Transactions on Graphics*, 2021. URL `http://www-sop.inria.fr/reves/Basilic/2021/PMGD21`. 30, 63, 110

Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18 (6):311–317, jun 1975. ISSN 0001-0782. doi: 10.1145/360825.360839. URL https://doi.org/10.1145/360825.360839. 22, 23

Pierre Poulin and Alain Fournier. A model for anisotropic reflection. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, page 273–282, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0897913442. doi: 10.1145/97879.97909. URL https://doi.org/10.1145/97879.97909. 23

Siddhant Prakash, Thomas Leimkühler, Simon Rodriguez, and George Drettakis. Hybrid image-based rendering for free-view synthesis. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(1), apr 2021. doi: 10.1145/3451260. URL https://doi.org/10.1145/3451260. 8

Siddhant Prakash, Gilles Rainer, Adrien Bousseau, and George Drettakis. Deep scene-scale material estimation from multi-view indoor captures. *Computers & Graphics*, 2022. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2022.09.010. URL https://www.sciencedirect.com/science/article/pii/S0097849322001789. 8

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. Neural btf compression and interpolation. *Computer Graphics Forum*, 38(2):235–244, 2019. doi: https://doi.org/10.1111/cgf.13633. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13633. 22, 28

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. Unified neural encoding of btfs. *Computer Graphics Forum (Proceedings of Eurographics)*, 39(2), June 2020. doi: 10.1111/cgf.13921. 22, 28

Gilles Rainer, Adrien Bousseau, Tobias Ritschel, and George Drettakis. Neural precomputed radiance transfer. *Computer Graphics Forum*, 41(2):365–378, 2022. doi: https://doi.org/10.1111/cgf.14480. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14480. 30, 109

Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 117–128, New York, NY, USA, 2001. Association

for Computing Machinery. ISBN 158113374X. doi: 10.1145/383259.383271. URL https://doi.org/10.1145/383259.383271. 3, 19

Capturing Reality. Realitycapture reconstruction software. https://www.capturingreality.com/Product, 2018. 37, 38, 44, 51, 56, 65, 84, 86, 115, 116

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 18

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. Global illumination with radiance regression functions. *ACM Trans. Graph.*, 32(4), jul 2013. ISSN 0730-0301. doi: 10.1145/2461912.2462009. URL https://doi.org/10.1145/2461912.2462009. 29

Gernot Riegler and Vladlen Koltun. Free view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 623–640, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58529-7. 17, 39, 51, 52, 114

Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12216–12225, June 2021. 17

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10912–10922, October 2021. 64, 95

Simon Rodriguez, Thomas Leimkühler, Siddhant Prakash, Chris Wyman, Peter Shirley, and George Drettakis. Glossy probe reprojection for interactive global illumination. *ACM Trans. Graph.*, 39(6), nov 2020a. ISSN 0730-0301. doi: 10.1145/3414685.3417823. URL https://doi.org/10.1145/3414685.3417823. 109, 111

Simon Rodriguez, Siddhant Prakash, Peter Hedman, and George Drettakis. Image-based rendering of cars using semantic labels and approximate reflection flow. *Proc. ACM*

Comput. Graph. Interact. Tech., 3(1), April 2020b. doi: 10.1145/3384535. URL `https://doi.org/10.1145/3384535`. 4, 15, 18, 38, 43

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. 68

Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 501–518, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9. 4, 38, 44

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2016. 4

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 96

Burr Settles. Active learning literature survey. 2009. 93, 96

Peter S. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. PhD thesis, USA, 1991. UMI Order NO. GAX91-24487. 2, 11, 18

Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-based rendering*. Springer Science & Business Media, 2008. 3, 34

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)*, 31(4):100–1, 2012. 4, 15, 38

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Niessner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 17, 39

Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325, 2021. 18

Bruce Smith. Geometrical shadowing of a random rough surface. *IEEE transactions on antennas and propagation*, 15(5):668–671, 1967. 23

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 835–846, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933646. doi: 10.1145/1179352.1141964. URL https://doi.org/10.1145/1179352.1141964. 13, 14, 33

Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008. 13

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 94

Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3, 17, 39

Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 30, 63, 110

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. URL http://arxiv.org/abs/1906.05797. 26, 94

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015. 88

Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, DAC '64, page 6.329–6.346, New York, NY, USA, 1964. Association for Computing Machinery. ISBN 9781450379328. doi: 10.1145/800265.810742. URL https://doi.org/10.1145/800265.810742. 1

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. Neural BRDF representation and importance sampling. *Computer Graphics Forum*, 40(6):332–346, September 2021. doi: https://doi.org/10.1111/cgf.14335. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14335. 111

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the art on neural rendering. *Computer Graphics Forum*, 39(2):701–727, 2020. doi: https://doi.org/10.1111/cgf.14022. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14022. 3, 16, 34, 38

Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering, 2021. URL https://arxiv.org/abs/2111.05849. 3, 24

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. IGNOR: image-guided neural object rendering. *CoRR*, abs/1811.10720, 2018. URL http://arxiv.org/abs/1811.10720. 16, 38

Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323035. URL https://doi.org/10.1145/3306346.3323035. 17, 31, 39

K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces*. *J. Opt. Soc. Am.*, 57(9):1105–1114, Sep 1967. doi: 10.1364/JOSA.57.001105. URL http://opg.optica.org/abstract.cfm?URI=josa-57-9-1105. 25

Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 11, 18

Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995a. 11

Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 419–428, New York, NY, USA, 1995b. Association for Computing Machinery. ISBN 0897917014. doi: 10.1145/218380.218498. URL https://doi.org/10.1145/218380.218498. 2, 11

Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 65–76, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0897918967. doi: 10.1145/258734.258775. URL https://doi.org/10.1145/258734.258775. 2, 11

Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 836–850, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. 37, 44

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195–206, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673524. 22, 23, 24

Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605588582. doi: 10.1145/1661412.1618479. URL https://doi.org/10.1145/1661412.1618479. 70

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, June 2021. 3, 17

Gregory J. Ward. Measuring and modeling anisotropic reflection. *SIGGRAPH Comput. Graph.*, 26(2):265–272, jul 1992. ISSN 0097-8930. doi: 10.1145/142920.134078. URL https://doi.org/10.1145/142920.134078. 22, 23, 25

Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, jun 1980. ISSN 0001-0782. doi: 10.1145/358876.358882. URL https://doi.org/10.1145/358876.358882. 9, 10

Hongzhi Wu, Zhaotian Wang, and Kun Zhou. Simultaneous localization and appearance estimation with a consumer rgb-d camera. *IEEE Transactions on Visualization and Computer Graphics*, 22(8):2012–2023, 2016. doi: 10.1109/TVCG.2015.2498617. 26

Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. Scalable neural indoor scene rendering. *ACM Trans. Graph.*, 41(4), jul 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530153. URL https://doi.org/10.1145/3528223.3530153. 16

Jiamin Xu, Xiuchao Wu, Zihan Zhu, Qixing Huang, Yin Yang, Hujun Bao, and Weiwei Xu. Scalable image-based indoor scene rendering with reflections. *ACM Trans. Graph.*, 40 (4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459849. URL https://doi.org/10.1145/3450626.3459849. 15

Wenjie Ye, Yue Dong, Pieter Peers, and Baining Guo. Deep reflectance scanning: Recovering spatially-varying material appearance from a flash-lit video sequence. *Computer Graphics Forum*, 2021. doi: https://doi.org/10.1111/cgf.14387. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14387. 29, 63

Mulin Yu and Florent Lafarge. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, US, 2022. 84

Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 215–224, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311559. URL https://doi.org/10.1145/311535.311559. 19, 26, 60

Cem Yuksel, Sylvain Lefebvre, and Marco Tarini. Rethinking texture mapping. In *Computer Graphics Forum*, volume 38, pages 535–551. Wiley Online Library, 2019. 87

Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016. 3, 20, 37

Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5453–5462, June 2021a. 30, 63

Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *arXiv preprint arXiv:2106.01970*, 2021b. 30, 63

Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014. 37

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images, 2018. 3, 17, 39

Xilong Zhou and Nima Khademi Kalantari. Adversarial single-image svbrdf estimation with hybrid training. *Computer Graphics Forum*, 40(2):315–325, 2021. doi: https://doi.org/10.1111/cgf.142635. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.142635. 27, 65

Jia-Jie Zhu and José Bento. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956*, 2017. 96

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 111