UNIVERSITY OF CALIFORNIA SAN DIEGO

Appearance Acquisition for Digital 3D Content Creation

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

in

Computer Science

by

Sai Bi

Committee in charge:

Professor Ravi Ramamoorthi, Chair
Professor Manmohan Chandraker
Professor David Kriegman
Professor Hao Su
Professor Zhuowen Tu

2021

The Dissertation of Sai Bi is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2021

DEDICATION

To my family.

TABLE OF CONTENTS

Dissertation Approval Page iii

Dedication iv

Table of Contents v

List of Figures vii

List of Tables x

Acknowledgements xi

Vita xiv

Abstract of the Dissertation xv

Chapter 1 Introduction 1

Chapter 2 High-Quality Texture Maps from Inaccurate RGB-D Reconstructions 5
2.1 Introduction 5
2.2 Related Works 7
2.3 Algorithm 10
2.3.1 Patch-Based Energy Function 12
2.3.2 Optimization 16
2.4 Implementation Details 21
2.5 Results 22
2.6 Other Applications 26
2.7 Conclusion 31

Chapter 3 Geometry and Reflectance from Sparse Multi-View Images 33
3.1 Introduction 33
3.2 Related Works 36
3.3 Algorithm 37
3.3.1 Multi-View Depth Prediction 38
3.3.2 Multi-View Reflectance Prediction 40
3.3.3 Geometry Reconstruction 42
3.3.4 SVBRDF and Geometry Refinement 42
3.4 Implementation and Results 45
3.4.1 Evaluation on Synthetic Data 47
3.4.2 Evaluation on Real Captured Data 47
3.5 Conclusion 52

Chapter 4 Deep Reflectance Volumes for Relightable Appearance Acquisition 53
4.1 Introduction 53

4.2 Related Works . . . 55
4.3 Rendering with Deep Reflectance Volumes . . . 57
4.3.1 Volume rendering overview . . . 58
4.3.2 A discretized, differentiable volume rendering module . . . 59
4.4 Learning Deep Reflectance Volumes . . . 61
4.4.1 Overview . . . 61
4.4.2 Network architecture . . . 62
4.4.3 Loss function and training details . . . 63
4.5 Results . . . 64
4.6 Conclusion . . . 71

Chapter 5 Deep Relightable Appearance Models for Animatable Faces . . . 73
5.1 Introduction . . . 73
5.2 Related Works . . . 76
5.3 Data Acquisition . . . 79
5.4 Building Relightable Avatars . . . 81
5.4.1 DRAM_ℓ: A Late-conditioned Model . . . 83
5.4.2 DRAM_ε: An Early-conditioned Model . . . 87
5.5 Animating Relightable Avatars . . . 90
5.6 Results . . . 91
5.6.1 Evaluation of late-conditioning model . . . 91
5.6.2 Evaluation of early-conditioned models . . . 99
5.6.3 Animation from headset mounted cameras . . . 103
5.6.4 Limitations . . . 103
5.7 Conclusion . . . 104

Chapter 6 Conclusion and Future Work . . . 105

Appendix A Appendix for Chapter 1 . . . 109

Appendix B Appendix for Chapter 2 . . . 111
B.1 BRDF Model . . . 111
B.2 Comparison on Geometry Reconstruction . . . 112
B.3 Additional Ablation Study . . . 112

Bibliography . . . 115

LIST OF FIGURES

Figure 1.1. Our contributions in this dissertation. 3

Figure 2.1. An overview of our texture mapping results. 6

Figure 2.2. Comparison to previous methods on synthetic data. 6

Figure 2.3. An illustration of the texture mapping process 9

Figure 2.4. Our method can handle cases with missing geometries by synthesizing the missing content. 13

Figure 2.5. The effect of enforcing the two terms in our energy function. 16

Figure 2.6. An overview of our two-step optimization framework. 17

Figure 2.7. Comparison between results with single iteration and multiple iterations of search and vote. 20

Figure 2.8. Comparison between results with single-scale and multi-scale optimizations 22

Figure 2.9. Comparison against state-of-the-art methods. 23

Figure 2.10. Estimated geometry for the objects in Fig. 2.9. 24

Figure 2.11. Limitations of our method. 26

Figure 2.12. Hole-filling results with our method. 27

Figure 2.13. Texture reshuffling results with our method. 29

Figure 2.14. Multiview camouflage with our method. 31

Figure 3.1. An overview of our results. 34

Figure 3.2. Our acquisition setup and framework. 35

Figure 3.3. Our multi-view SVBRDF estimation network. 39

Figure 3.4. Comparison on geometries with our method and COLMAP. 48

Figure 3.5. Comparison on SVBRDF optimization. 49

Figure 3.6. Comparison with Nam et al. [97] on geometry refinement. 50

Figure 3.7. Comparison with Nam et al. [97] on rendered images. 50
Figure 3.8. Our results on real scenes. 51
Figure 4.1. A visualization of our deep reflectance volumes. 54
Figure 4.2. An overview of our framework for learning deep reflectance volumes. 59
Figure 4.3. Comparisons with mesh-based reconstruction. 66
Figure 4.4. Additional results on real scenes. 68
Figure 4.5. Comparison between our deep prior based optimization and direct optimization of the volume without using networks. 70
Figure 4.6. Material editing results with our method. 71
Figure 5.1. Rendering results with our deep relightable appearance model. 74
Figure 5.2. Our acquisition system. 80
Figure 5.3. Network architecture for our late-conditioned model. 83
Figure 5.4. Network architecture for our early-conditioned model. 90
Figure 5.5. Comparisons between our predicted OLAT images and the ground truth. . 92
Figure 5.6. Late-conditioned model: rendering under novel *directional lights*. 93
Figure 5.7. Renderings under environment maps with our late-conditioned model. . . . 93
Figure 5.8. Nearfield relighting with our late-conditioned model. 94
Figure 5.9. Comparison on Results with and without depth differences. 95
Figure 5.10. Visual comparisons between renderings on novel expressions with our late-conditioned model trained on static captures and dynamic captures. . . 96
Figure 5.11. Late-conditioned model: a visual comparison of different capture lighting patterns. 97
Figure 5.12. Visual comparison between our early-conditioned and late-conditioned model. 100
Figure 5.13. Visual comparison between our method and the state-of-the-art method. . . 100

Figure 5.14. Visual comparison between our hyper-network architecture and baseline models. 101

Figure 5.15. Our early-conditioned model under novel environment maps animated by VR headset mounted cameras. 102

Figure B.1. Comparison with Nam et al. [97] on geometry optimization. 112

Figure B.2. Comparison between optimizations with and without per-view warping. . . 113

Figure B.3. Comparison on results with and without per-vertex refinement. 113

LIST OF TABLES

Table 2.1. Notation used in this chapter. 11

Table 3.1. Quantitative SVBRDF evaluation on a synthetic test set against state-of-the-art methods. 47

Table 4.1. Performance of our method with different number of training images. 67

Table 4.2. Comparison against DeepVoxels on synthetic data. 67

Table 5.1. Feature comparison with previous methods. 76

Table 5.2. Quantitative comparison on results with and without depth difference. 97

Table 5.3. Quantitative comparison on models trained on captures under different lighting patterns 98

Table 5.4. Late-conditioned model: comparison between static captures and dynamic captures. 98

Table 5.5. Comparison between our early-conditioned model with hyper-networks and baseline models. 99

Table B.1. Quantitative comparisons between networks trained with different inputs on the synthetic test set. 112

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof. Ravi Ramamoorthi for his patient guidance, insightful suggestions and endless support during my Ph.D. journey. It has been a privilege of mine to work with and learn from Ravi. During the past years, Ravi has provided invaluable instructions to me in terms of determining research topics, developing novel solutions, polishing academic writings and delivering good presentations, which have helped me become more independent in research and will bring me lifelong benefits.

I also want to thank Prof. Manmohan Chandraker, Prof. David Kriegman, Prof. Hao Su, Prof. Zhuowen Tu for serving on my thesis committee. They have provided helpful feedback and suggestions on my work. Their input helps make the thesis better. I would also like to thank my undergraduate advisor Prof. Yizhou Yu. I got to know computer graphics from his course at The University of Hong Kong, and he also offered me an opportunity to work with him in his group, which makes me get a first sense of what doing research is like and lays the foundation for my study at UC San Diego.

I've been fortunate to be able to work with many wonderful collaborators. This thesis cannot be completed without their great help. Thanks to Nima Kalantari, Ravi Ramamoorthi, Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, Hao Su, Federico Perazzi, Eli Shechtman, Vladimir Kim, David Kriegman, Miloš Hašan, Yannick Hold-Geoffroy, Pratul Srinivasan, Ben Mildenhall, Stephen Lombardi, Shunsuke Saito, Tomas Simon, Shih-En Wei, Kevyn McPhail, Yaser Sheikh, Jason Saragih, Manmohan Chandraker and Zhengqin Li. I enjoy the collaboration with these talented researchers, and I am grateful for their constructive feedback and comments on the projects. In particular, I want to thank Nima for hands-on help and valuable suggestions at the beginning of my Ph.D. Kalyan has provided many insightful suggestions and detailed comments on many projects, which always enlightens and inspires me when I got stuck with some problems. Zexiang has been a close collaborator and friend, and we had many fruitful discussions on technical details and brainstormed abundant interesting ideas.

During my Ph.D., I've been lucky to do internships at different companies, and I am

greatly grateful to my mentors, including Shengyang Dai (Google), Baoyuan Wang (Microsoft Research), Kalyan Sunkavalli (Adobe Research) and Jason Saragih (Facebook Reality Labs). They have provided me great opportunities to get to know the exciting research and projects done at the companies. They proposed interesting projects for me and gave valuable guidance during the internship. Great thanks to them for their support and mentorship.

Apart from doing research, the life during my Ph.D. has been fantastic thanks to many wonderful friends. Thanks to my lab mates at Ravi's group for making the lab a collaborative and inspiring environment. I would also like to thank my friends Yao Qin, Mengting Wan, Lifan Wu, Zexiang Xu, Zhengqin Li, Weilun Sun, Lingqi Yan, Tiancheng Sun and Yixing Lao for their support in both life and work.

Finally, I would like to thank my parents and my brother for their endless love and support. They are the inspiration for me to go further, and they are the comfort when I meet with a hurdle. This thesis is dedicated to them.

This work was supported in part by ONR grants N000141512013, N000141712687, N000141912293, NSF grants 1451830, 1617234, Adobe, the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing.

Chapter 2 is based on the material as it appears in ACM Transactions on Graphics, 2017 ("Patch-Based Optimization for Image-Based Texture Mapping", Sai Bi, Nima Khademi Kalantari, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 3 is based on the material as it appears in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019 ("Deep 3D Capture: Geometry and Reflectance from Sparse Multi-View Images", Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 4 is based on the material as it appears in European Conference on Computer Vision (ECCV), 2020 ("Deep Reflectance Volumes: Relightable Reconstructions from Multi-View Photometric Images", Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick

Hold-Geoffroy, David Kriegman, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 5 is based on the material as submitted to ACM Transactions on Graphics, 2021 ("Deep Relightable Appearance Models for Animatable Faces", Sai Bi, Stephen Lombardi, Shunsuke Saito, Tomas Simon, Shih-En Wei, Kevyn McPhail, Ravi Ramamoorthi, Yaser Sheikh, Jason Saragih). The dissertation author was the primary investigator and author of this paper.

VITA

2011–2014 B.Eng in Computer Science, The University of Hong Kong

2015–2019 M.S. in Computer Science, University of California San Diego

2015–2021 Ph.D. in Computer Science, University of California San Diego

ABSTRACT OF THE DISSERTATION

Appearance Acquisition for Digital 3D Content Creation

by

Sai Bi

Doctor of Philosophy in Computer Science

University of California San Diego, 2021

Professor Ravi Ramamoorthi, Chair

Digital 3D content plays an important role in many applications such as video games, animations, virtual reality (VR) and augmented reality (AR). Traditionally, the creation of digital 3D content requires complicated software and designers with special expertise, which imposes great challenges for novice users. In comparison, an alternative approach for this task is to acquire and digitize the appearance of real-world scenes by capturing images and automatically reconstructing 3D models from the scans. In this dissertation, we present a series of works for efficiently and accurately creating 3D representations from the captured images. We exploit appropriate representations for both scene geometry and reflectance to support different functionalities including novel view synthesis, relighting and dynamic animations.

First, we present an approach for generating accurate texture maps for RGB-D reconstructions that enable us to navigate the scene under novel viewpoints. Our method can correct misalignments between captured images caused by inaccurate camera poses and corrupted geometries and produce highly-quality texture maps. Afterwards we take one step further and propose a learning-based method to reconstruct high-quality meshes with per-vertex BRDFs from a sparse set of six images captured under collocated camera and light, which supports visualization of the scene under novel viewpoints and lighting conditions. Then we go beyond traditional mesh representations and propose to learn a novel volumetric representation that encodes volume density, normal and reflectance properties at any arbitrary 3D point in the scene for joint view synthesis and relighting. We demonstrate that our volumetric representation can be estimated from images captured with a simple collocated camera-light setup, and accurately model the appearance of real-world scenes with complex geometry and reflectance. Finally, we develop approaches for modeling the dynamic appearance of human faces and learning animatable lifelike avatars that support free-viewpoint relighting and novel expressions. We apply neural networks to directly regress the facial geometry and textures under the desired viewpoints, lightings and expressions. We show that our model can be animated and driven by images captured with VR-headset mounted cameras, demonstrating the first system for face-driven interactions in VR that uses a photorealistic relightable face model.

Chapter 1

Introduction

With the advent of new applications such as VR and AR, the demand for digital 3D content has been increasing in an unprecedented way. Traditionally we rely on experienced designers to create 3D content using professional software, which requires special expertise and highly time-consuming manual efforts. In contrast, real-world objects and scenes have provided us an abundant source for 3D content. Therefore, acquiring and digitizing the appearance of real-world objects and scenes has been a core task in computer vision and graphics. In this dissertation, we propose a series of methods to automatically reconstruct high-quality 3D models from the captured images of real-world objects and scenes.

The appearance of real-world scenes depends on two components including geometry and materials, both of which have various representations. Geometry can be represented using depth maps, point clouds, triangle meshes and volumes etc. More recently, researchers have also applied neural networks to represent the geometry using implicit signed distance functions [104] and implicit volumetric representations [94]. In terms of materials, for diffuse objects, we can simply represent their appearance using 2D texture maps. For non-Lambertian objects, bidirectional reflectance distribution functions (BRDFs) are commonly used to model how light is reflected on the surface of the objects. We exploit appropriate scene representations to support different functionality including changing viewpoints, relighting and dynamic animations. We also propose effective methods to automatically and accurately reconstruct these representations

from the input images.

Specifically, in Chapter 2, we propose a novel method to produce accurate texture maps for geometric models of real-world objects represented with triangle meshes. Although a high-quality texture map can be easily computed for accurate geometry and calibrated cameras, the quality of texture map degrades significantly in the presence of inaccuracies. In this work, we address this problem by proposing a novel global patch-based optimization system to synthesize the aligned images. Specifically, we use patch-based synthesis to reconstruct a set of photometrically-consistent aligned images by drawing information from the source images. Our optimization system is simple, flexible, and more suitable for correcting large misalignments than other techniques such as local warping. To solve the optimization, we propose a two-step approach which involves patch search and vote, and reconstruction. Our approach can produce high-quality texture maps (Figure 1.1a) better than existing techniques for objects scanned by consumer depth cameras such as Intel RealSense. We demonstrate that our system can be used for texture editing tasks such as hole-filling and reshuffling as well as multi-view camouflage.

Texture maps are only suitable to model the appearance of diffuse objects and do not support relighting under novel lighting conditions. However, most real-world objects are non-Lambertian, and their appearance changes depending on the illumination conditions. Therefore, in Chapter 3, we introduce a novel learning-based method to reconstruct the high-quality geometry and complex, spatially-varying BRDFs (Figure 1.1b) of an arbitrary object. Unlike previous approaches that rely on high-end 3D scanners or dense input images, we achieve this using a sparse set of only six images We first estimate per-view depth maps using a deep multi-view stereo network; these depth maps are used to coarsely align the different views. We propose a novel multi-view reflectance estimation network architecture that is trained to pool features from these coarsely aligned images and predict per-view spatially-varying diffuse albedo, surface normals, specular roughness and specular albedo. Finally, we fuse and refine these per-view estimates to construct high-quality geometry and per-vertex BRDFs. We do this by jointly optimizing the latent space of our multi-view reflectance network to minimize the

(a) Chapter 2: high-quality texture maps from inaccurate RGB-D reconstructions.

(b) Chapter 3: geometry and reflectance from sparse multi-view images.

(c) Chapter 4: neural reflectance volumes for relightable appearance acquisition.

(d) Chapter 5: deep relightable appearance models for animatable faces.

Figure 1.1. Our contributions in this dissertation. We propose novel methods to reconstruct different representations for objects and scenes to faithfully reproduce their appearance.

photometric error between images rendered with our predictions and the input images. While previous state-of-the-art methods fail on such sparse acquisition setups, we demonstrate that our method produces high-quality reconstructions that can be used to render photorealistic images.

In the first two works, we have been using triangle meshes to represent the geometry of the captured objects. While triangle meshes have many advantages and are widely used, it is usually difficult to accurately reconstruct them for complex scenes with thin structures, heavy occlusions and non-convex shapes such as trees. Their irregular structure also makes it difficult to be applied in an end-to-end learning pipeline. In contrast, volumes have regular 3D structures and can be easily integrated with neural networks. Therefore, in Chapter 4, we introduce a novel volumetric scene representation that encodes volume density, normal and reflectance properties

at any arbitrary 3D point in the scene. We combine this representation with a physically-based differentiable ray marching framework that can render images under any viewpoint and light. We demonstrate that our volumetric representation can be estimated from images captured with a simple collocated camera-light setup, and accurately model the appearance of real-world scenes with complex geometry and reflectance. This allows us to perform high-quality view synthesis and relighting (Figure 1.1c) that is significantly better than previous methods.

Different from previous chapters that focus on reproducing the appearance of static objects, in Chapter 5 we present a method for building high fidelity dynamic animatable 3D face models (Figure 1.1d) that can be posed and rendered with novel lighting environments in real-time. Human faces have complex appearance including diffuse and specular reflections and subsurface scattering, which cannot be accurately modeled by analytical BRDF models based on physical priors as those used in previous chapters. Therefore, in this chapter we apply neural networks to model the complex reflectance of human faces. Our model takes in the expression code, view direction and lighting condition as input and directly regresses the geometry for the current expression and the view-light specific texture. Our method is capable of capturing subtle lighting effects and can even generate compelling near-field relighting despite being trained exclusively with far-field lighting data. We motivate the utility of our model by animating it from VR-headset mounted cameras, demonstrating the first system for face-driven interactions in VR that uses a photorealistic relightable face model.

Finally, in Chapter 6, we summarize the dissertation and discuss some future works on related topics such as appearance acquisition in an unconstrained environment, capturing objects and scenes with complex light transport effects, neural rendering for large-scale scenes and neural representations from sparse images.

Chapter 2

High-Quality Texture Maps from Inaccurate RGB-D Reconstructions

2.1 Introduction

With the availability of consumer depth cameras to the public, ordinary users are now able to produce geometric models of objects using techniques like KinectFusion [98]. However, reproducing the full appearance of real-world objects also requires reconstructing high-quality texture maps. Image-based texture mapping is a common approach to produce a view-independent texture map from a set of images taken from different viewpoints. However, this is a challenging problem since the geometry and camera poses are usually estimated from noisy data, and thus, are inaccurate. Moreover, the RGB images from consumer depth cameras typically suffer from optical distortions which are not accounted for by the camera model. Therefore, naïvely projecting and combining the input images produces blurring and ghosting artifacts, as shown in Fig. 2.2.

We observe that we can overcome most of the inaccuracies by generating an aligned image for every input image. Our method builds upon the recent work by Zhou and Koltun [161] that proposes an optimization system to correct the misalignments using local warping. Although this approach handles small inaccuracies, it fails to produce high-quality results in cases with large inaccuracies and missing geometric features because of the limited ability of local warping in correcting misalignments (see Figs. 2.1, 2.2 and 2.4).

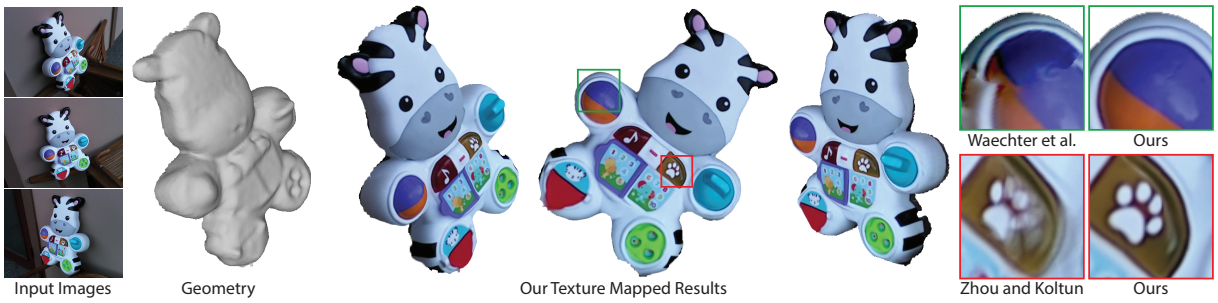

Figure 2.1. The goal of our approach is to produce a high-quality texture map given the geometry of an object as well as a set of input images and their corresponding camera poses. A small subset of our input images as well as the rough geometry, obtained using the KinectFusion algorithm, are shown on the left. Since the estimated geometry and camera poses are usually inaccurate, simply projecting the input images onto the geometry and blending them produces unsatisfactory results with ghosting and blurring artifacts. We handle the inaccuracies of the capturing process by proposing a novel patch-based optimization system to synthesize aligned images. Here, we show different views of an object rendered using OpenGL with the texture map generated using our system. Our approach produces high-quality texture maps and outperforms state-of-the-art methods of Waechter et al. [135] and Zhou and Koltun [161].

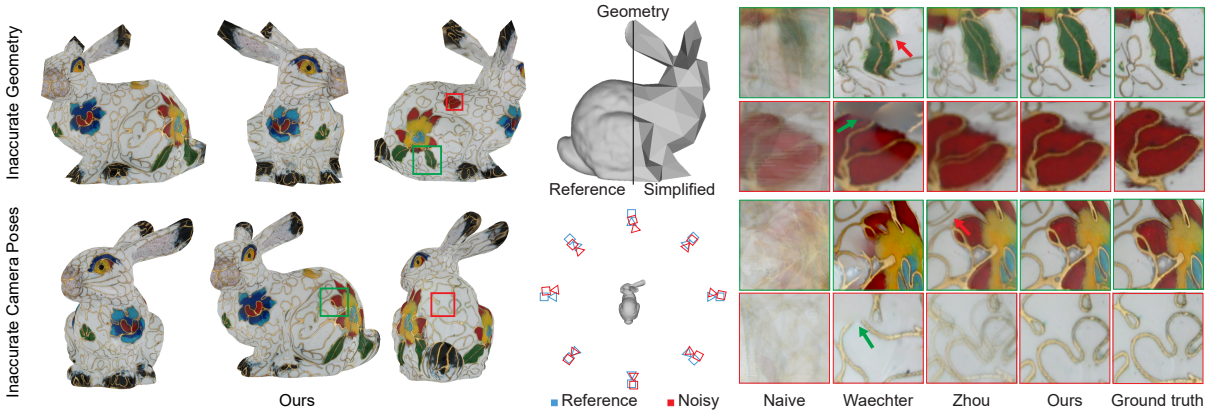

Figure 2.2. We generate 24 input views by rendering a synthetic textured bunny from different viewpoints and artificially add inaccuracies by simplifying the geometry and adding noise to the camera poses. We compare our approach against state-of-the-art methods as well as naïvely projecting and combining the images. Waechter et al.'s approach [135] selects a single view per face by solving a complex optimization system to reduce the artifacts around the face boundaries. However, their results contain visible seams because of large inaccuracies in the geometry and camera poses. Zhou and Koltun [161] tackle the inaccuracies of the geometry by using local warping to align the input images, but fail to properly register the images and produce unsatisfactory results. Moreover, when the camera poses are significantly inaccurate, their system converges to a local minimum, and thus, their results suffer from ghosting and blurring artifacts. Our approach is more flexible and can properly synthesize aligned images that when combined produce artifact-free texture in both cases.

Inspired by the recent success of patch-based methods in image and video editing tasks, we propose a novel global patch-based optimization system to *synthesize* aligned images. Our energy function combines our two main desirable properties for the aligned images; **1)** include *most* of the information from the original input images, and **2)** preserve the photometric consistency of the projection. By optimizing our proposed energy function, we simultaneously maximize the local similarity of the aligned and input images and ensure the consistency of all the aligned images and the texture map.

Our system draws information from the source images in a patch-based manner, and thus, is flexible and able to handle large inaccuracies. Moreover, our method handles cases with missing geometric features (see Fig. 2.4) by synthesizing the missing content, while the existing warping-based [161] and graph-cut based [135] techniques are not able to do so. Finally, in contrast to Zhou and Koltun's approach, we perform the optimization in the image domain which makes the performance of our system independent of the complexity of the geometry. In summary, we make the following contributions:

- We introduce the first patch-based optimization system for view-independent image based texture mapping (Sec. 2.3.1). Our method corrects misalignments by synthesizing aligned images which can then be used to produce a single view-independent texture map.
- We propose a simple iterative two-step approach to efficiently solve our energy equation (Sec. 2.3.2).
- We show that our approach produces better results than existing techniques (Sec. 2.5). Furthermore, we show other applications of our system (e.g., texture hole-filling) which are not possible to do with the current methods.

2.2 Related Works

Reproducing the full appearance of a real-world object from a set of images has been the subject of extensive research. Image-based rendering approaches [17, 51] reproduce the

appearance of an object by generating a view-dependent texture map [31]. However, these methods are only able to provide the ability to navigate an object with the lighting condition of the input photographs. Therefore, they cannot be used for applications where the goal is to use the scanned object in a new environment with different lightings. Moreover, since these approaches do not produce a globally consistent texture map, they are typically not used in gaming, augmented reality, and animations.

View-independent texture mapping approaches like our own, produce a single consistent texture map from a set of images captured from different viewpoints, which can then be rendered with different lightings.[1] The main challenge of these methods is addressing the inaccuracies in the capturing process. Several methods have been presented to register the images to the geometry in a semi-automatic way [102, 108, 38] or automatically by, for example, optimizing color consistency [109, 13], aligning image and geometric features [78, 126], and maximizing the mutual information between the projected images [28, 27]. While these methods are effective at addressing the camera calibration inaccuracies, they are not able to handle inaccurate geometry, and optical distortions in RGB images which are common problems of consumer depth cameras.

A small number of approaches have been proposed to tackle general inaccuracies. We categorize these approaches in two classes and discuss them in the following two subsections.

Single View Selection – Instead of blending the projected input images, which could generate blurry results because of misalignments, these approaches select only one view per face. To avoid visible seams between the boundaries of each face, they typically solve a discrete labeling problem [134, 77, 123, 135].

For example, the state-of-the-art method of Waechter et al. [135] solves a conditional random field energy equation, consisting of two terms: a data term which favors views that are closer to the current face and are less blurry, and a smoothness term which penalizes inconsistencies between adjacent faces. However, as shown in Fig. 2.2, even this approach is not

[1] Note that, the final textures in this case still have the original lighting condition. However, this problem can be addressed by applying intrinsic decomposition [15] on the source images and using albedo to generate the texture maps.

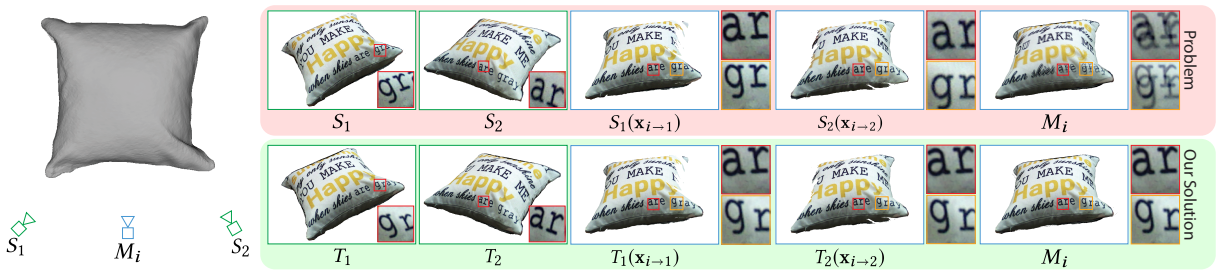

Figure 2.3. Here, the goal is produce a high-quality texture map, using the rough geometry as well as two source views, shown on the left. We illustrate the texture mapping process from a novel view, shown with the blue camera. Because of the inaccuracies in the geometry, camera poses, and optical distortions of the source images, the projected source images to view i, $S_1(\mathbf{x}_{i\to1})$ and $S_2(\mathbf{x}_{i\to2})$, are typically misaligned. For example, the inset shows that the position of "ar" and "gr" in the two projected source images is different. Therefore, combining (averaging in this case) these projected source images produces a texture map, M_i, with blurring and ghosting artifacts. We propose to handle this misalignment problem by synthesizing a target image for every source image in a way that the projected target images are photometrically consistent. To reconstruct each target image, we keep the overall visual appearance of the source image, but move its content to correct the misalignments. Note the difference in position of "gr" and "ar" in the source and target images. In this case, since the projected images are aligned, we are able to produce a ghost-free high-quality texture map.

able to handle the large inaccuracies in challenging cases, producing visible seams in the final texture map.

Image Alignment – The approaches in this category directly handle the inaccuracies by aligning the input images. Tzur and Tal [133] propose to estimate local camera projection for each vertex of the geometry to handle inaccuracies from calibration, geometry, etc. However, their approach requires user interaction to produce plausible results. Aganj et al. [2] address misalignment by finding matching SIFT features in different views and warping the input images, while others [36, 32] perform warping using optical flow. These methods do not minimize the distortion globally and work on a pair of images, and thus, are sub-optimal. Gal et al. [41] assigns each triangle to one input image and finds the optimum shift for each triangle to remove the seams, but their optimization is computationally expensive.

The recent work of Zhou and Koltun [161], which our method builds upon, solves an optimization system to find optimum camera poses as well as non-rigid corrections of the input

images, simultaneously. They use local warping to perform non-rigid alignment and propose an alternating optimization to minimize their objective function. However, the local warping is not able to correct large misalignments and it produces results with ghosting and blurring artifacts in challenging cases, as shown in Fig. 2.2. To avoid this problem, we propose a different optimization system with a more flexible mechanism for non-rigid alignment than local warping.

Patch-Based Synthesis – Our approach is inspired by the recent success of patch-based synthesis methods in a variety of applications such as hole-filling [140], image retargeting and editing [122, 10], morphing [120], HDR reconstruction [118, 61], and style transfer [11, 58]. Patch-based synthesis has been shown to be particularly successful in applications where finding correspondences between two or multiple images (e.g., morphing and HDR reconstruction) is difficult. In our application, the synthesized aligned images need to be consistent with respect to the object's geometry, and thus, direct application of patch-based synthesis to our problem does not work. We address this challenge by proposing a novel patch-based energy equation which incorporates the geometry into the formulation.

2.3 Algorithm

The goal of most image-based texture mapping approaches is to produce a high-quality view-independent texture map using a set of N source images, $S_1, \cdots, S_N$, taken from different viewpoints. These methods usually assume that the object's approximate geometry and rough camera poses (i.e., extrinsic and intrinsic parameters) of all the source images are already estimated using existing techniques [117, 98]. Once the texture map is created, the object with a view-independent texture can be rendered from any novel views.

A simple way to produce a texture map is to project the source images onto the geometry and combine all the projected images. Ideally, these projected images are photometrically consistent, and thus, combining them produces a high-quality texture map. However, in practice, because of the inaccuracies, the projected images are typically misaligned. Therefore, this simple

Table 2.1. Notation used in this chapter.

$S_1, \ldots, S_N$	source images (input)
$T_1, \ldots, T_N$	target (aligned) images (output)
$M_1, \ldots, M_N$	texture at different views (output)
$\mathbf{x}_i$	pixel position on image i
$\mathbf{x}_{i \to j}$	pixel position projected from image i to j
$T_j(\mathbf{x}_{i \to j})$	RGB color of the j^{th} target image at pixel $\mathbf{x}_{i \to j}$, i.e., the result of projecting target j to camera i

approach produces texture maps with ghosting artifacts.

We show this problem in Fig. 2.3 (top row) for a case with two source images S_1 and S_2. To observe the misalignment problem, we project the source images to a novel view i. Note that, projection from a source image S_j to a novel view i can be performed by remapping the source image's pixel colors, $S_j(\mathbf{y})$. Here, $\mathbf{y}$ is the projection of the pixels from image i to j. Formally, we can write this as:

$$\mathbf{y} = \mathscr{P}_j(\mathscr{G}_i(\mathbf{x})),$$

where $\mathbf{x}$ is the pixel position on image i, $\mathscr{G}_i$ projects a pixel on image i to the global 3D space, and $\mathscr{P}_j$ projects a 3D point to the image j. In this chapter, for clarity and simplicity of the notation, we use $\mathbf{x}_i$ and $\mathbf{x}_{i \to j}$ to denote the pixels on image i and the pixels projected from image i to j, respectively. In this case, $\mathbf{y} = \mathbf{x}_{i \to j}$ and $S_j(\mathbf{x}_{i \to j})$ is the result of projecting source image S_j to view i. See Table 2.1 for the complete list of notation used in this chapter.

As shown in Fig. 2.3 (top row), because of the inaccuracies in the estimated geometry and camera poses, the projected source images, $S_1(\mathbf{x}_{i \to 1})$ and $S_2(\mathbf{x}_{i \to 2})$, are misaligned. Therefore, the texture map generated by the simple projection and blending approach contains ghosting artifacts (rightmost column). Here, M_i refers to the final globally consistent texture map, seen from camera i. Note that, M_j is reconstructed from all the source images, and thus, is different from the projected source images.

To overcome this misalignment problem, we propose to *synthesize* an aligned (target) image, T_i, for every source image, S_i. As shown in Fig. 2.3, the targets are reconstructed by moving the content of the source images to correct the misalignment. As a result, all the target images are photometrically consistent, and thus, projecting them onto the geometry and combining them produces a high-quality result. In the next section, we explain our patch-based optimization system to synthesize these target images.

2.3.1 Patch-Based Energy Function

Our main observation is that to produce a high-quality texture map, the target images should have two main properties: **1)** each target image should be similar to its corresponding source image, and **2)** the projected target images should be photometrically consistent. Our goal is to propose a global energy function which codifies these two main properties.

To satisfy the first property we ensure that each target image contains *most* of the information from its corresponding source image in a visually coherent way. To do so, we use bidirectional similarity (BDS) as proposed by Simakov et al. [122]. This is a patch-based energy function which is defined as:

$$E_{\text{BDS}}(S,T) = \frac{1}{L}(\underbrace{\sum_{s \subset S} \min_{t \subset T} D(s,t)}_{\text{completeness}} + \alpha \underbrace{\sum_{t \subset T} \min_{s \subset S} D(s,t)}_{\text{coherence}}), \tag{2.1}$$

where α is a parameter defining the ratio of these two terms, s and t are patches from the source S and target T images respectively, and D is the sum of squared differences of all the pixel values of the patches s and t in RGB color space. Moreover, L is the number of pixels in each patch, e.g., $L = 49$ for a 7×7 patch.

Here, the first term (completeness) ensures that every source patch has a similar patch in the target and vice versa for the second term (coherence). The completeness term measures how much information from the source is included in the target, while the coherence term measures if there are any new visual structures (artifacts) in the target image. Minimizing this energy

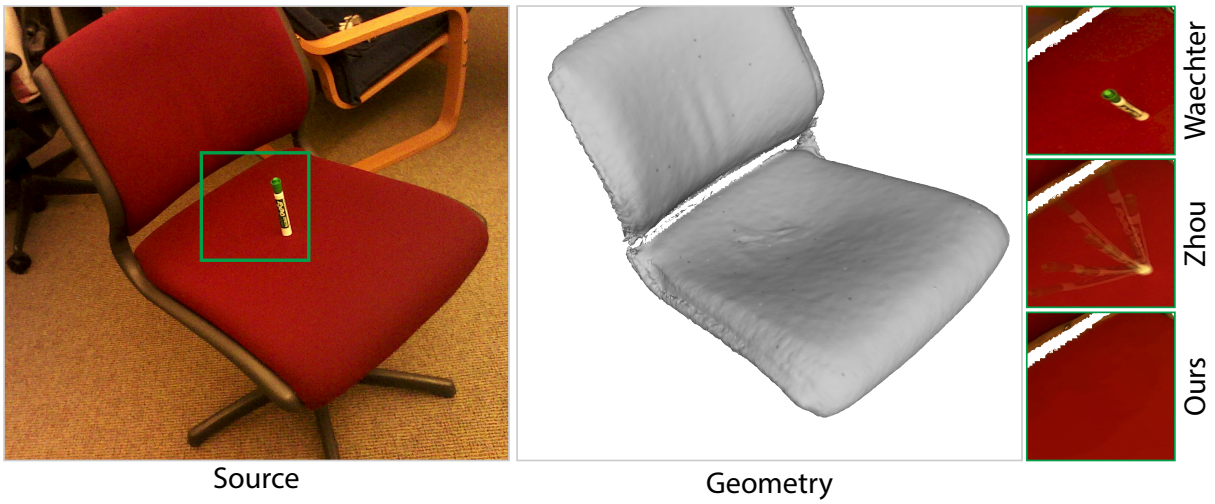

Figure 2.4. Here, we have a scene with a chair and a green marker on top as can be seen in the source image. Because of the inaccuracies of the consumer depth camera, the marker's geometry is not reconstructed. In this case, the texture from the marker should not appear in the final texture map. The method of Waechter et al. [135] selects one of the source images for each triangle in the geometry. Since the marker exists in all the source images, this method incorrectly places it in the final texture. Zhou and Koltun [161] align the source images by locally warping them. Therefore, they are not able to remove the marker from the aligned images, resulting in ghosting artifacts. Our patch-based approach synthesizes the target images, and thus, only includes valid information from the source images. Therefore, we are able to remove the marker from the source images and produce an artifact-free result.

function ensures that most of the information from the source is included in the target image in a visually coherent way. In our implementation, we set $\alpha = 2$ to give more importance to the coherence term.

Note that, Eq. 2.1 is defined for a single pair of source and target images. To enforce the similarity property for all the images, we extend this equation as:

$$E_1 = \sum_{i=1}^{N} E_{\mathrm{BDS}}(S_i, T_i). \tag{2.2}$$

Patch-based synthesis is more flexible than local warping [161], and thus, is more suitable to handle large inaccuracies in the geometry and the camera poses. Furthermore, while local warping inherently preserves the visual coherency, it includes *all* the information from the source in the aligned (target) image which is not desirable in our application. If the geometric model

does not contain specific features, the regions corresponding to these features should not be included from the source images in the texture map. Therefore, this method produces results with blurring and ghosting artifacts in these regions, as shown in Fig. 2.4. Waechter et al.'s method [135] selects one view per face and can avoid ghosting artifacts in this case. However, this approach is not able to remove the texture corresponding to the missing feature, since it exists in all the source images. Note that, missing geometric features occur in most cases with significantly inaccurate geometry (Fig. 2.9), which is why the existing techniques poorly handle these challenging cases.

Although the similarity of the target and source images is a necessary condition for generating a high-quality texture map, it is not sufficient, as shown in Fig. 2.5. Therefore, we need to enforce the second property by ensuring the consistency of the target images. This constraint can be implemented in several ways. For example, we can enforce the consistency by ensuring that the projected target images are close to the current target, i.e., $T_j(\mathbf{x}_{i\to j}) = T_i(\mathbf{x}_i)$. This constraint can be formally written as the ℓ^2 distance between $T_j(\mathbf{x}_{i\to j})$ and $T_i(\mathbf{x}_i)$ and be minimized in a least square sense.

Alternatively, the constraint can be enforced by ensuring the consistency of the current target and average of all the projected targets, i.e., $1/N\sum_{j=1}^{N} T_j(\mathbf{x}_{i\to j}) = T_i(\mathbf{x}_i)$. Similarly, we can enforce the texture at view i to be consistent with the projected target images, i.e., $T_j(\mathbf{x}_{i\to j}) = M_i(\mathbf{x}_i)$, to enforce the constraint. Since all the target images will be consistent with each other and the final texture map after optimization, these different approaches result in similar optimum target images. However, to be able to utilize alternating optimization (see Sec. 2.3.2), we use the last strategy ($T_j(\mathbf{x}_{i\to j}) = M_i(\mathbf{x}_i)$) and write our consistency energy equation as:

$$E_C(\{T_j\}_{j=1}^{N}, M_i) = \frac{1}{N}\sum_{\mathbf{x}_i}\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j})\left(T_j(\mathbf{x}_{i\to j}) - M_i(\mathbf{x}_i)\right)^2, \tag{2.3}$$

where the first summation is over all the pixel positions $\mathbf{x}_i$ on image i. Here, the weight w_j enforces the constraint to be proportional to the contribution of the j^{th} projected target image. In

our implementation, $w_j = cos(\theta)^2/d^2$, where θ is the angle between the surface normal and the viewing direction at image j and d denotes the distance between the camera and the surface.[2] This weight basically gives smaller weight to the cameras that look at the surface at a grazing angle and are further away from the object. Minimizing this energy function ensures that all the target images are consistent with the final texture map viewed from camera i. We extend this equation to enforce the consistency constraint for all the images as:

$$E_2 = \sum_{i=1}^{N} E_C(\{T_j\}_{j=1}^{N}, M_i) \tag{2.4}$$

To satisfy our two properties, we propose the complete objective function to be the weighted summation of E_1 and E_2:

$$E = E_1 + \lambda E_2, \tag{2.5}$$

where λ defines the weight of the consistency term and we set it to 0.1 in our implementation. Optimizing our proposed patch-based energy function produces target images that contain most of the information from the source images, are visually coherent, and preserve the consistency of the projection. Once the optimum target images, T_i, are obtained, they can be used to produce a single consistent texture in different ways. For example, this can be done by first projecting all the target images to the geometry. After this process, each vertex receives a set of color samples from different target images. The final color of each vertex can then be obtained by computing the weighted average of these color samples.[3]

We evaluate the effect of each term in our optimization system in Fig. 2.5. Optimizing the first term alone produces aligned images that have the same visual appearance as the source images, but are not consistent. Optimizing the second term produces consistent target images, but they contain information that does not exist in the source images. Optimizing our proposed

[2] We use the interpolated normal and vertex from the fragment shader.

[3] We can generate the global texture with either the target images, T_i, or the textures, M_i, as they are very similar after optimization.

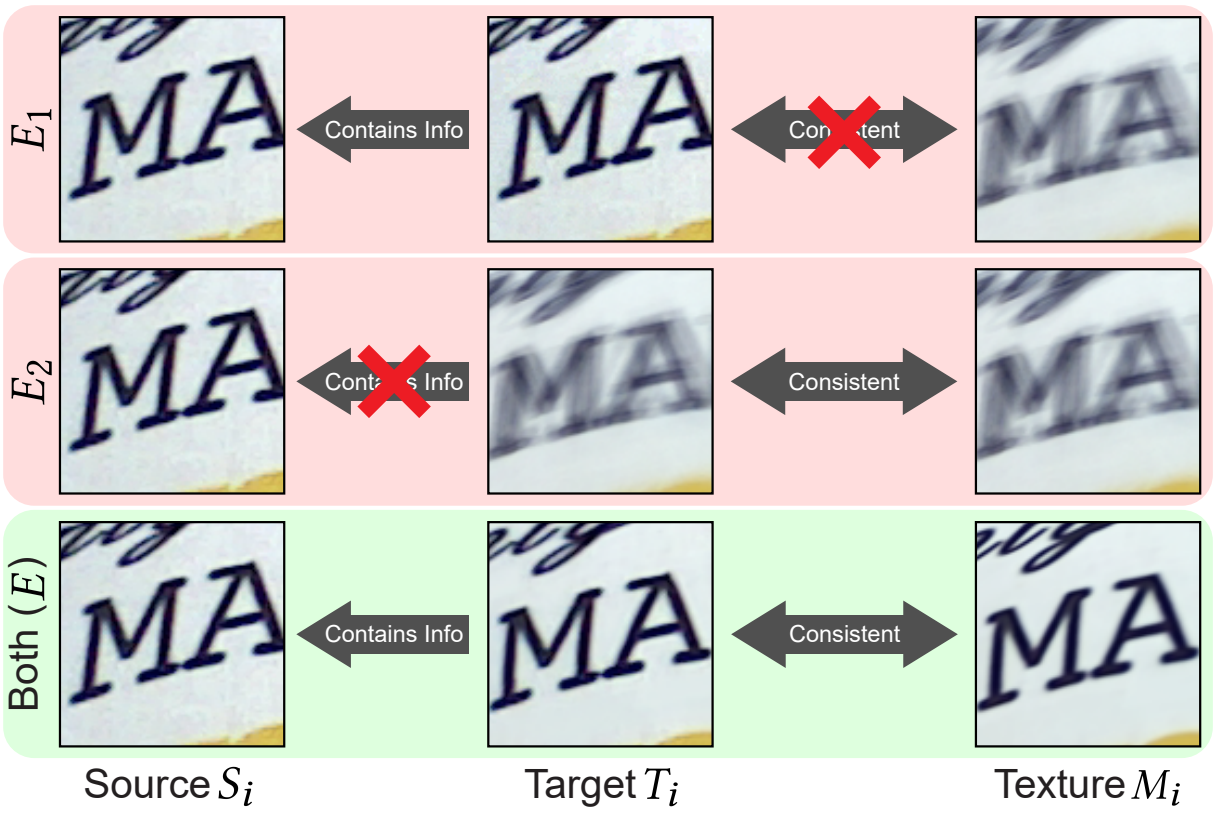

Figure 2.5. We evaluate the effect of enforcing our main properties with the two terms in Eq. 2.5. Only optimizing the first term ensures that the target image contains most of the information of the source image. However, since the consistency constraint is not enforced, the final texture is not consistent with the target image. On the other hand, by only optimizing the second term, the consistency constraint is enforced, and thus, the target image and the final texture are photometrically consistent. However, the target image has ghosted content which does not appear in the source image. Our full approach optimizes both terms and ensures that the targets contain source contents and are consistent. Therefore, only our full approach is able to produce a high-quality texture map.

full energy function produces a high-quality texture map by enforcing both properties.

2.3.2 Optimization

To efficiently optimize our energy function in Eq. 2.5, we propose an alternating optimization approach which simultaneously solves for the target images, $T_1, \cdots, T_N$, and the texture at different views, $M_1, \cdots, M_N$. Specifically, we minimize our energy function by alternating between optimizing our two sets of variables. We initialize the targets and textures with their

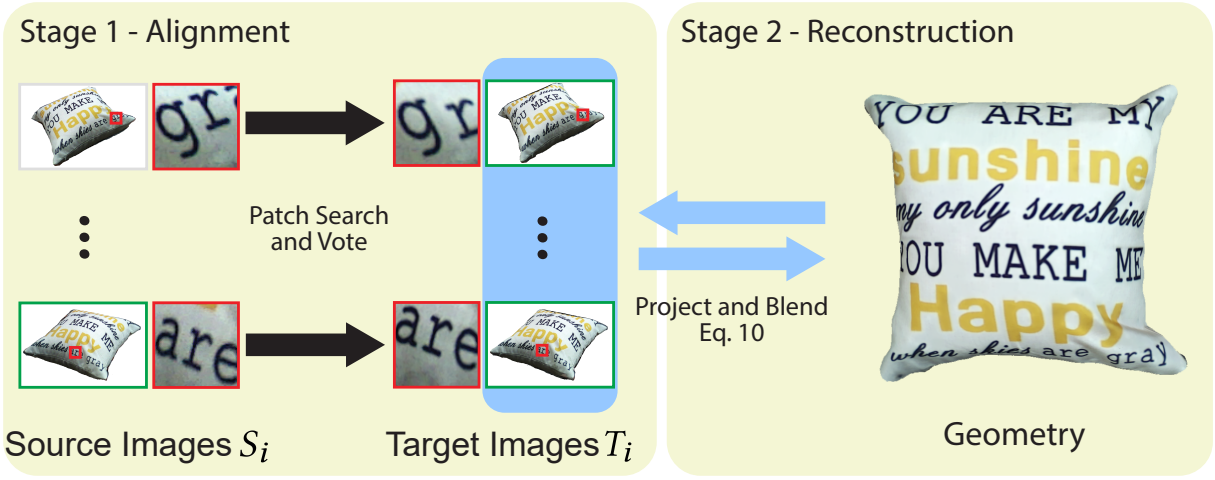

Figure 2.6. Our approach synthesizes aligned images (target) by optimizing Eq. 2.5. We propose to optimize this energy function with a two-step approach. During alignment, patch search and vote is performed between the source and target images to obtain new targets. Note that, while the source and target images are similar, the target image is reconstructed by moving the source content to ensure alignment. In the reconstruction stage, the target images are projected on the geometry and combined (Eq. 2.10) to produce the texture at different views. The two steps of alignment and reconstruction are continued iteratively and in multiple scales until convergence.

corresponding source images, i.e., $T_i = S_i$ and $M_i = S_i$. We then iteratively perform our two steps of alignment and reconstruction until convergence. The overview of our algorithm is given in Fig. 2.6. Below, we explain our two steps:

1) Alignment – In this stage, we fix $M_1, \cdots, M_N$ and minimize Eq. 2.5 by finding optimum $T_1, \cdots, T_N$. This is done using an iterative search and vote process similar to Simakov et al. [122]. In the first step, we perform a patch search process, as proposed by Simakov et al., to find the patches with minimum $D(s,t)$ (see Eq. 2.1), where D is the sum of squared differences. In the next step, we perform the voting process to obtain $T_1, \cdots, T_N$ that minimize Eq. 2.5 given the calculated patches in the previous step. Note that, as we will discuss next, there is a key difference between our and the original voting [122] which is because of our additional consistency constraint, E_C.

For the sake of clarity, we explain our voting by first discussing each term of Eq. 2.5 separately.

First (Similarity) Term: We start by rewriting the BDS energy function (E_1) using the

obtained patches during search as done in Simakov et al. [122]:

$$E_1(i,\mathbf{x}_i) = \frac{1}{L}\left[\sum_{u=1}^{U}\left(s_u(\mathbf{y}_u) - T_i(\mathbf{x}_i)\right)^2 + \alpha\sum_{v=1}^{V}\left(s_v(\mathbf{y}_v) - T_i(\mathbf{x}_i)\right)^2\right]. \tag{2.6}$$

where $E_1(i,\mathbf{x}_i)$ refers to the error E_1 for a specific camera i and pixel $\mathbf{x}_i$. Here, s_u and s_v are the source patches overlapping with pixel $\mathbf{x}_i$ of the target for the completeness and coherence terms, respectively. Moreover, $\mathbf{y}_u$ and $\mathbf{y}_v$ refer to a single pixel in s_u and s_v, respectively, corresponding to the $\mathbf{x}_i^{\text{th}}$ pixel of the target image. Finally, U and V refer to the number of patches for the completeness and coherence terms, respectively. Note that, most of these variables are a function of the current pixel, $\mathbf{x}_i$, but we omit this dependence for simplicity of the notation. See the original paper by Simakov et al. [122] for the derivation of this equation. To obtain T_i's that minimize the above equation, we need to differentiate the error with respect to the unknown color $T_i(\mathbf{x}_i)$ and set it equal to zero which results in:

$$T_i(\mathbf{x}_i) = \frac{\frac{1}{L}\sum_{u=1}^{U} s_u(\mathbf{y}_u) + \frac{\alpha}{L}\sum_{v=1}^{V} s_v(\mathbf{y}_v)}{\frac{U}{L} + \frac{\alpha V}{L}}. \tag{2.7}$$

Here, the target is obtained by computing a weighted average of the pixel colors of a set of source patches, overlapping with the $\mathbf{x}_i^{\text{th}}$ pixel of the target image. Note that, although the normalization terms, $1/L$, cancel out, we keep them here to be able to easily combine this equation with the next term (Eq. 2.8) in Eq. 2.9.

Second (Consistency) Term: The first term is the standard voting process, as proposed by Simakov et al., and basically draws information from the source image to reconstruct the targets. Our key difference lies in the second term which enforces the consistency constraint by ensuring that the target images are close to the textures. As shown in the Appendix, the targets

minimizing the second term in Eq. 2.5 can be calculated as:

$$T_i(\mathbf{x}_i) = \frac{\frac{1}{N} w_i(\mathbf{x}_i) \sum_{k=1}^{N} M_k(\mathbf{x}_{i \to k})}{w_i(\mathbf{x}_i)}. \tag{2.8}$$

Again, although the weights, $w_i(\mathbf{x}_i)$, cancel out, we keep them in this equation for clarity, when combining the two terms in Eq. 2.9. Here, each target is computed by averaging the current texture maps from different views. This is intuitive as the constraint basically enforces the aligned image to be as close as possible to the textures.

Combined Terms: Intuitively, the targets solving the combined terms should be reconstructed by drawing information from the source images, while staying similar to the textures. Since the two terms are combined with a λ factor (see Eq. 2.5), the combined solution can be computed by separately adding the numerator and denominator of the terms in Eqs. 2.7 and 2.8 as:

$$T_i(\mathbf{x}_i) = \frac{\frac{1}{L}\sum_{u=1}^{U} s_u(\mathbf{y}_u) + \frac{\alpha}{L}\sum_{v=1}^{V} s_v(\mathbf{y}_v) + \frac{\lambda}{N} w_i(\mathbf{x}_i) \sum_{k=1}^{N} M_k(\mathbf{x}_{i \to k})}{\frac{U}{L} + \frac{\alpha V}{L} + \lambda w_i(\mathbf{x}_i)}. \tag{2.9}$$

As can be seen, the final updated target is a weighted average of the result of regular voting (Eq. 2.7) and the average of all the current texture maps (Eq. 2.8). This means that the consistency term basically enforces our updated targets to remain close to the current textures.

This energy function is minimized by iteratively performing the search and vote process until convergence. These iterations work by using the updated targets after voting as the input to the search process in the next iteration. We empirically found that only one iteration of search and vote is sufficient to obtain high-quality results, as shown in Fig. 2.7.

2) Reconstruction – In this step, we fix $T_1, \cdots, T_N$ and produce optimum texture at different views, $M_1, \cdots, M_N$, to minimize Eq. 2.5. Since the textures only appear in the second

Figure 2.7. We show that a single iteration of search and vote produces results that are very similar to those with multiple iterations.

term (E_C), which is quadratic, the optimal textures can be easily obtained as follows:

$$M_i(\mathbf{x}_i) = \frac{\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j}) T_j(\mathbf{x}_{i\to j})}{\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j})}. \tag{2.10}$$

This is our texture generation equation which basically states that the optimum texture is obtained by computing a weighted average of all the projected targets. In case the targets are misaligned, which is usually the case at the beginning of the optimization, this process produces textures with ghosting and blurring. The next iteration of the alignment process will then try to reduce the misalignment between the targets, which consequently results in a texture map with fewer artifacts after reconstruction.

We continue this process of alignment and reconstruction iteratively until convergence. As is common with the patch-based approaches [140, 10], we perform this process at multiple scales to avoid local minima and speed up the convergence (see Sec. 2.4). Note that the iterations here are done between our two main stages of alignment and reconstruction. We also have an inner iteration between the search and vote process at every alignment stage. However, as discussed, we found that only one iteration of search and vote is sufficient during alignment.

Once converged, our algorithm produces the aligned images, $T_1, \cdots, T_N$, as well as the optimum texture at different views, $M_1, \cdots, M_N$, which will be very similar. Since our target images are consistent, a single global texture can be obtained by projecting all the target images

on the geometry and averaging their color samples to obtain the final color at each vertex.

2.4 Implementation Details

Capturing input data – We use an Intel RealSense R200 camera to capture our input RGB-D sequences. This camera records depth and color sequences with a resolution of 628×468 and 1920×1080, respectively, both at 30 fps. To minimize the color variations, we use fixed exposure and white balancing. We estimate the geometry and the camera poses of each frame using the KinectFusion algorithm [57]. Note that, this approach estimates the camera pose of the depth frames and we also assign these estimated camera poses to the corresponding color frames.[4]

Keyframe Selection – To reduce the number of our input images, we select a subset of images with a greedy approach similar to Zhou and Koltun's method [161]. Specifically, given a set of already selected key frames, we use the method of Crete et al. [29] to find a frame with the lowest blurriness in the interval of $(t, 2t)$ after the last selected key frame. In our implementation, t varies between 30 to 60 frames depending on the scene.

Alignment – To accelerate the search process, we use the PatchMatch algorithm of Barnes et al. [10] with the default parameters and patch size of 7. Moreover, to avoid the target images deviating significantly from the source images, we limit the search to a small window of size $0.1\sqrt{w \times h}$, where w and h are the width and height of the source image.

Multiscale Optimization – We solve our energy function in Eq. 2.5 by performing the optimization in multiple scales. Specifically, we start by downsampling all the source images to the coarsest scale. We first initialize the targets, $T_1, \cdots, T_N$, and the textures, $M_1, \cdots, M_N$, with the low resolution source images and perform the alignment and reconstruction stages iteratively until convergence. We then upsample all the targets and textures to the resolution of the next scale and perform our two stages iteratively at this new scale. Note that, instead of upsampling

[4] One may obtain the color camera poses by applying a rigid transformation to the depth camera poses, but this strategy would not significantly help for two reasons: 1) the shutters of the depth and color cameras are not perfectly synchronized, and 2) our depth and color cameras are close to each other, and thus, they have similar poses.

Figure 2.8. The energy function in Eq. 2.5 has a large number of local minima. By minimizing this energy function at the finest scale, there is a significant possibility of getting trapped in one of these local minima. Similar to other patch-based approaches, we perform the optimization at multiple scales to produce high-quality results, as shown on the right.

the sources from the coarser scale, we directly downsample the original high resolution source images to the current scale. This allows the system to inject high frequency details into the targets and textures. We continue this process for all the finer scales to obtain the final targets at the finest scale. In the coarsest scale, the input image has 64 pixels in the smaller dimension and we have a total of 10 scales with scaling factor of $\sqrt[9]{x/64}$, where x is the smaller dimension of the original source images. We perform 50 iterations of alignment and reconstruction at the coarsest scale and decrease it by 5 at each finer scale.

As shown in Fig. 2.8, this multiscale approach is necessary to avoid local minima, and consequently, produce high-quality results. Intuitively, our optimization system aligns the global structures in the coarser scales and recovers the details in the finer scales. A video demonstrating the convergence of our algorithm at multiple scales can be found in the supplementary material of the original publication.

2.5 Results

We implemented our framework in MATLAB/C++ and compared against the state-of-the-art approaches by Eisemann et al. [36], Waechter et al. [135] and Zhou and Koltun [161]. We used the authors' code for Waechter et al. and Eisemann et al.'s approaches, but implemented

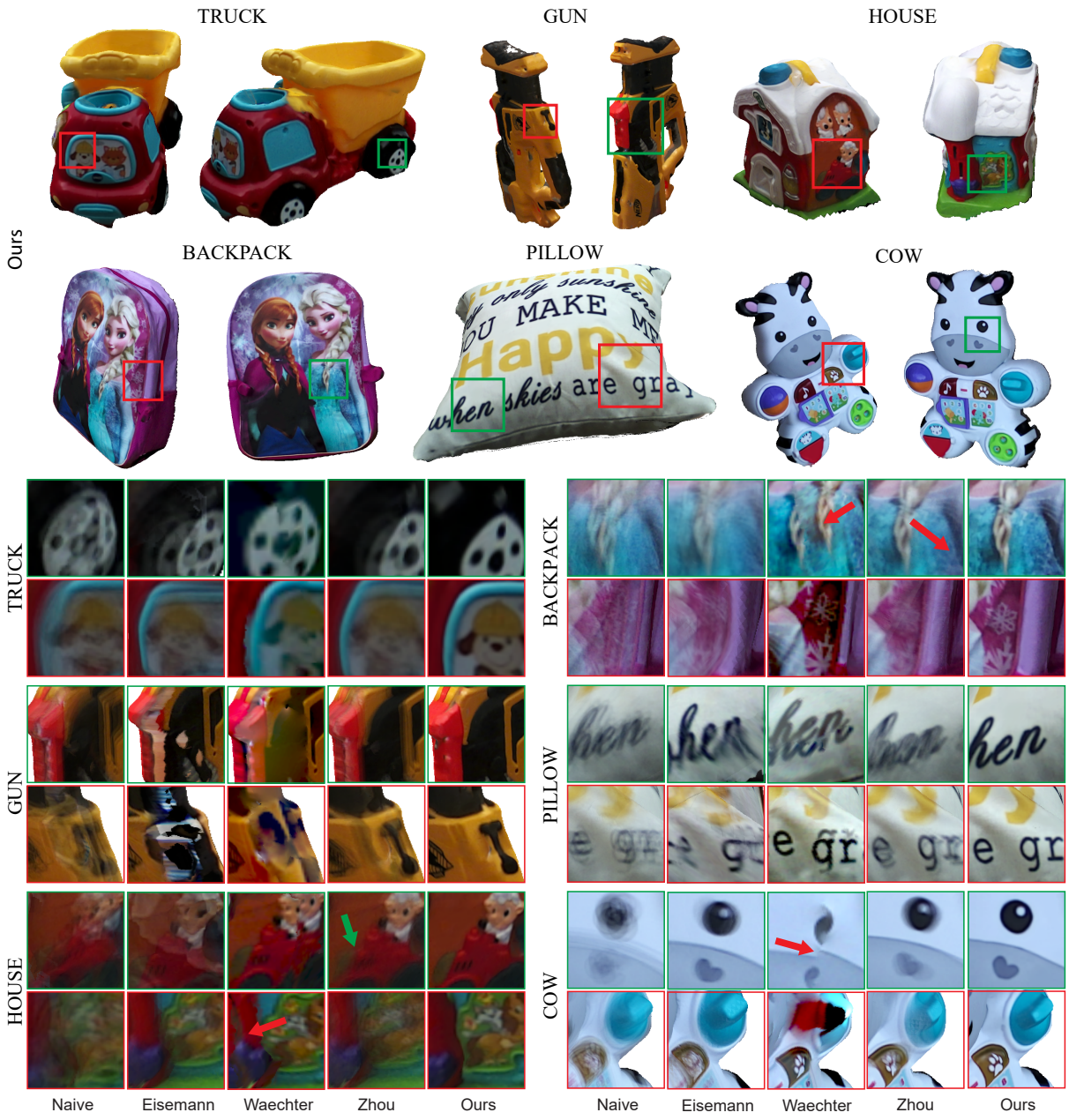

Figure 2.9. We compare our approach against the state-of-the-art algorithms of Eisemann et al. [36], Waechter et al. [135] and Zhou and Koltun [161]. We also demonstrate the result of naïvely projecting all the images and averaging them for comparison. Other approaches are not able to handle these challenging scenes and produce results with tearing, discoloration, blurring, and ghosting artifacts. On the other hand, we generate artifact-free high-quality results.

the method of Zhou and Koltun ourself since their source code is not available online. Note that, for Eisemann et al.'s approach, we use the implementation for static scenes and generate

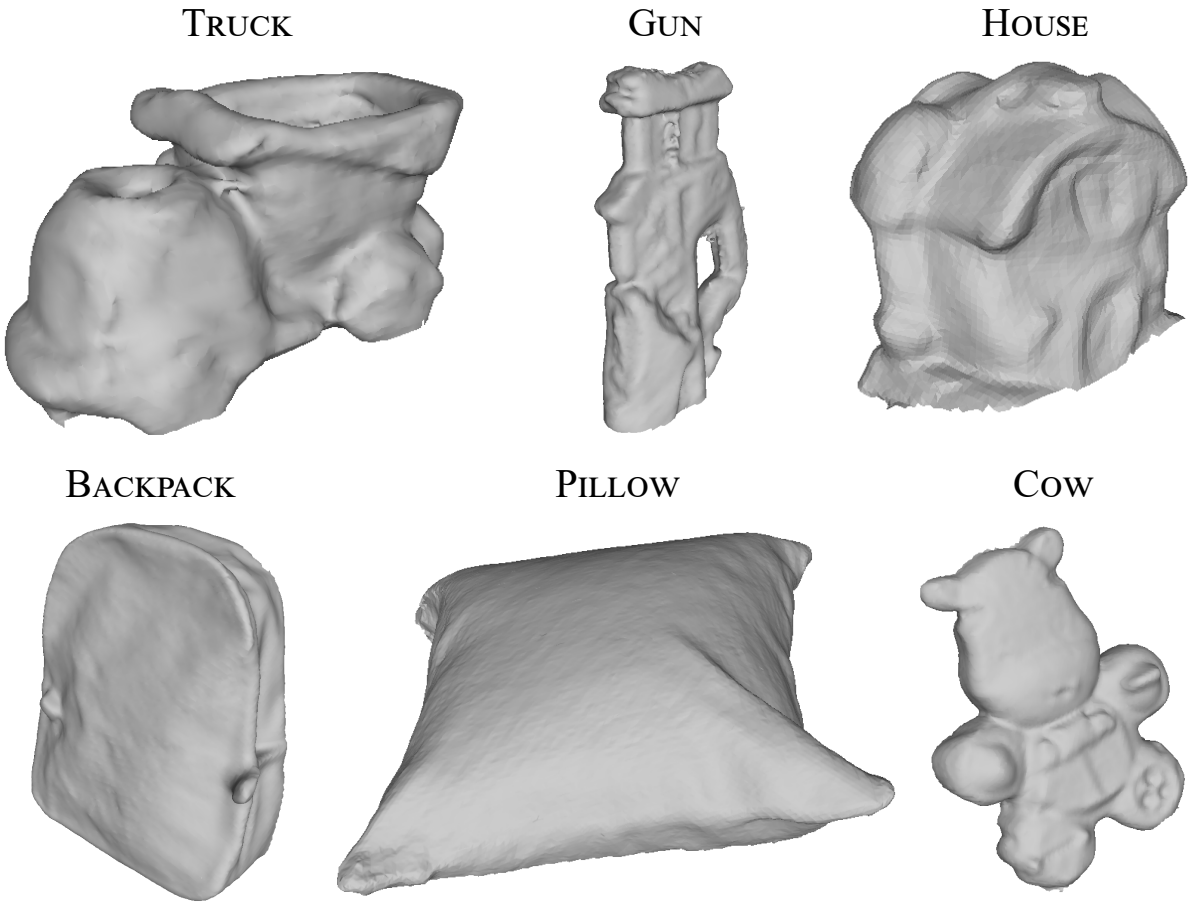

Figure 2.10. Estimated geometry for the objects in Fig. 2.9.

view-independent textures to have a fair comparison. We demonstrate the results by showing one or two views of each object, and videos showing the texture mapped objects from different views can be found in the supplementary video of the original publication. Note that, our scenes are generally more challenging than Zhou and Koltun's scenes. This is mainly because of the fact that we casually capture our scenes under typical lighting conditions, and thus, our geometries have lower accuracy. We have tested our method on the FOUNTAIN scene from Zhou and Koltun's paper and are able to produce comparable results, as shown in Fig. 2.13 (Aligned Target).

Figure 2.9 compares our approach against other methods on six challenging objects, and the estimated geometry for these objects is shown in Fig. 2.10. The TRUCK is a challenging scene with a complex geometry which cannot be accurately captured with consumer depth

cameras. Eisemann et al. [36] works on a pair of images and corrects misalignments using optical flow without optimizing a global energy function, which is suboptimal. Therefore, their method produces blurry textures as their warped images typically contain residual misalignments. Waechter et al. [135] select one view per face by solving an optimization system to hide the seams between adjacent faces. However, their method is not able to produce satisfactory results in this case, since they assign inconsistent textures to some of the adjacent faces because of significant inaccuracies. Note the tearing artifacts at the top inset and the distorted bear face at the bottom inset. Moreover, the local warping in Zhou and Koltun's approach [161] is not able to correct significant misalignment in this case, caused by inaccurate geometry (see Fig. 2.10). Therefore, their results suffer from ghosting and blurring artifacts. Our method synthesizes aligned target images and is able to produce high-quality texture maps with minimal artifacts.

None of the other approaches are able to handle the GUN scene. Specifically, note that only our approach is able to reconstruct the thin black structure at the bottom inset. Because of inaccuracies in optical flow estimation, Eisemann et al.'s approach produces results with tearing artifacts. It is worth noting that the method of Waechter et al. performs color correction to fix the color variations between adjacent faces. Since in this case the images are significantly misaligned, adjacent faces may have inconsistent textures. Therefore, the color correction introduces discoloration which is visible in the two insets. Next, we examine the HOUSE scene, which has a complex geometry. Waechter et al. produce tearing artifacts, while Eisemann et al. and Zhou and Koltun's results demonstrate ghosting artifacts. This is mainly due to the complexity of this scene and the inaccuracy of the geometry (see Fig. 2.10). On the other hand, our method is able to produce high-quality results on this challenging scene.

The top inset of the BACKPACK scene shows a region with a fairly smooth geometry. However, Eisemann et al.'s method is still not able to properly align the images and generates blurry textures. Moreover, Waechter et al.'s method generates results with tearing artifacts due to incorrect camera poses. Although Zhou and Koltun's method corrects most of the misalignments in this case, their result is slightly blurrier than ours. The bottom inset shows a region from

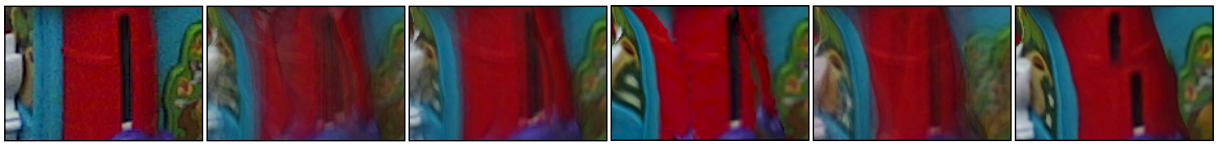

Figure 2.11. We show a small inset on the side of the toy house in the HOUSE scene (see Fig. 2.9). The input images are significantly misaligned as can be seen by the blurriness of the Naïve approach. Our method is able to correct the misalignments and produce a plausible result. However, our patch-based approach is not able to always preserve the semantic information. For example, our method produces a result, where the single hole is broken down into two separate pieces. Other approaches are able to produce results with a single hole, but they suffer from tearing and blurring artifacts.

the side of the backpack with a complex geometry. In this region, Waechter et al.'s method demonstrates discoloration artifacts, while Zhou and Koltun and Eisemann et al.'s approaches produce results with ghosting artifacts. Similarly, none of the other methods are able to properly reconstruct the textures on the sides of the PILLOW, a region with complex geometry. It is worth mentioning that Waechter et al.'s approach also produces discoloration artifacts in the underside of the pillow which can be seen in the supplementary video of the original publication. Finally, only our method is able to properly reconstruct the eye and heart at the top inset and the blue and brown structures at the bottom insets of the COW scene.

Limitation – The main limitation of our approach is that patch-based synthesis generally produces plausible results, but in some cases is not able to preserve the semantic information, as shown in Fig. 2.11. Here, although our approach corrects the significant misalignments and produces plausible results, it is unable to preserve the structure of the hole (see the source inset) and breaks it down into two segments.

2.6 Other Applications

In this section, we discuss several applications of our patch-based system including texture hole-filling and reshuffling as well as multiview camouflage. Note that, although patch-based synthesis has been previously used for *image* hole-filling and reshuffling [122, 10], these

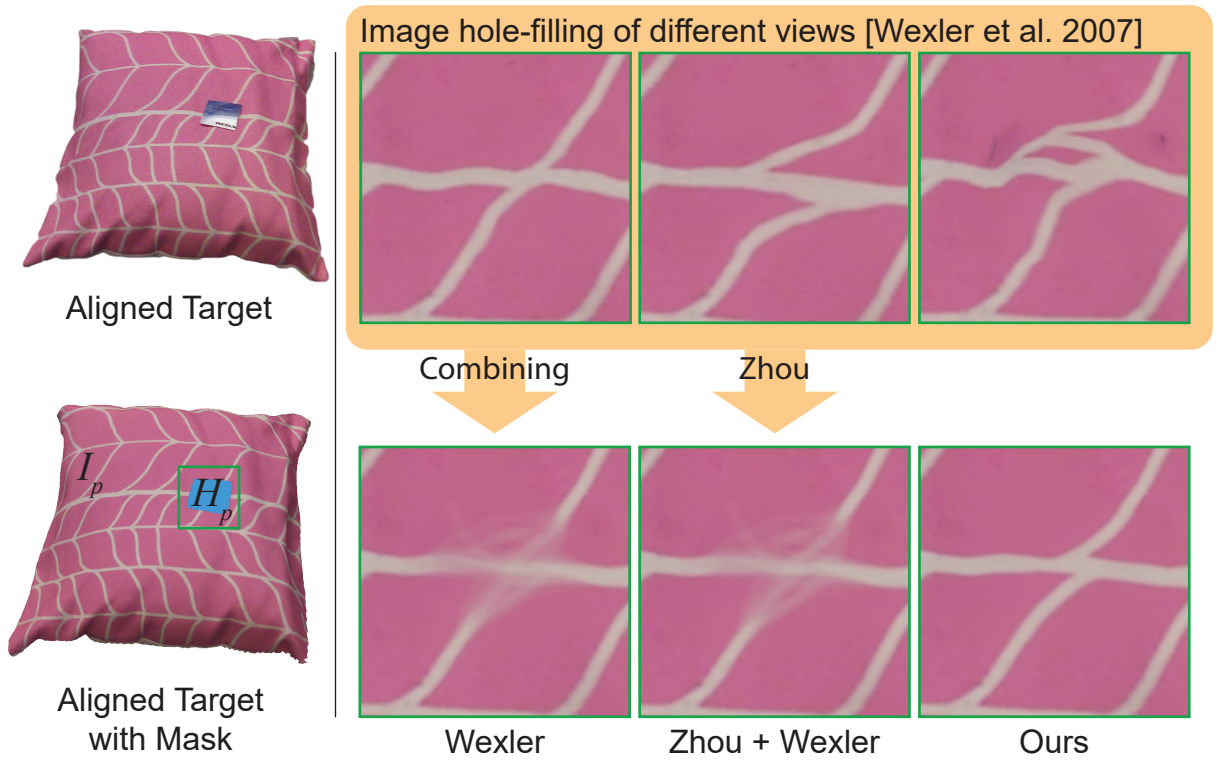

Figure 2.12. We first use our system to synthesize aligned target images, one of which shown on the top left. We then mark the unwanted region (the sticker on the pillow) as the hole and project it to all the other views to obtain holes in other targets. The top row shows three views of the hole-filled results using the traditional patch-based synthesis [140] to fill in the hole at each target image independently (we project the results to the same view for better comparison). Although the hole-filled results at each view are plausible, combining them produces texture with ghosting artifacts because of their inconsistencies. Aligning the hole-filled images using the method of Zhou and Koltun [161] only slightly reduces the blurriness. Our method completes the holes in different targets in a photometrically consistent way, and thus, is able to produce artifact-free results.

methods are not suitable in our application because of lack of consistency.

Texture Hole-filling – In some cases, the texture of a real-world object may contain unwanted regions (holes) that we wish to fill in. One example of this case is shown in Fig. 2.12, where the sticker on the pillow is not desired and should be removed from the final texture map. To do so, we begin by synthesizing aligned target images using our system. We then mark the hole region (shown in blue) in one of the aligned target images. This region can be simply projected to the other views to generate the hole in all the targets. These marked regions basically

divide each target image into hole H_i and input I_i (the region outside the hole).

Here, the goal is to fill in the holes, H_i, by drawing information from each input, I_i, while preserving the photometric consistency of the filled holes. This is very similar to the main properties of our energy function in Eq. 2.5, and thus, our system can be used to perform the hole-filling process. Note that, this problem is related to multi-view hole-filling which has been proposed in a few recent techniques [8, 131], but we present a way to perform this task using our texture mapping framework.

We do this by setting the sources to the inputs, $S_i = I_i$, and the targets to the holes, $T_i = H_i$ in Eq. 2.5. In this case, our optimization draws information from the sources (regions outside the holes) to fill in the targets (holes) in a consistent way. This is done by performing the patch search from the regions outside the hole to the holes and voting these patches to reconstruct only the hole regions. For initialization, instead of using the sources, we smoothly fill in the holes from the boundary pixels using MATLAB's roifill function. We also omit the completeness term in the BDS energy term (see Eq. 2.1) which is responsible for bringing *most* of the information from the source to the target images. Note that, while this is a requirement for alignment, it is not necessary for hole-filling since we only need partial information from the inputs to fill in the holes.

We compare our approach to patch-based image hole-filling [140] in Fig. 2.12. Although performing the hole-filing separately can produce plausible results at each view (top row), combining them generates a texture with ghosting artifacts (bottom row - left) because of their inconsistency. The method of Zhou and Koltun [161] can be used to align the hole-filled images at different views (bottom row - middle). However, the final texture still contains ghosting artifacts since the inconsistencies cannot be corrected with warping. Our method is able to produce consistent hole-filled results in different views, and consequently, generate high-quality hole-filled texture.

It is worth noting that we do not hole-fill the geometry. Therefore, our method can only fill in texture holes, if their underlying geometry is not complex, like the one in Fig. 2.12.

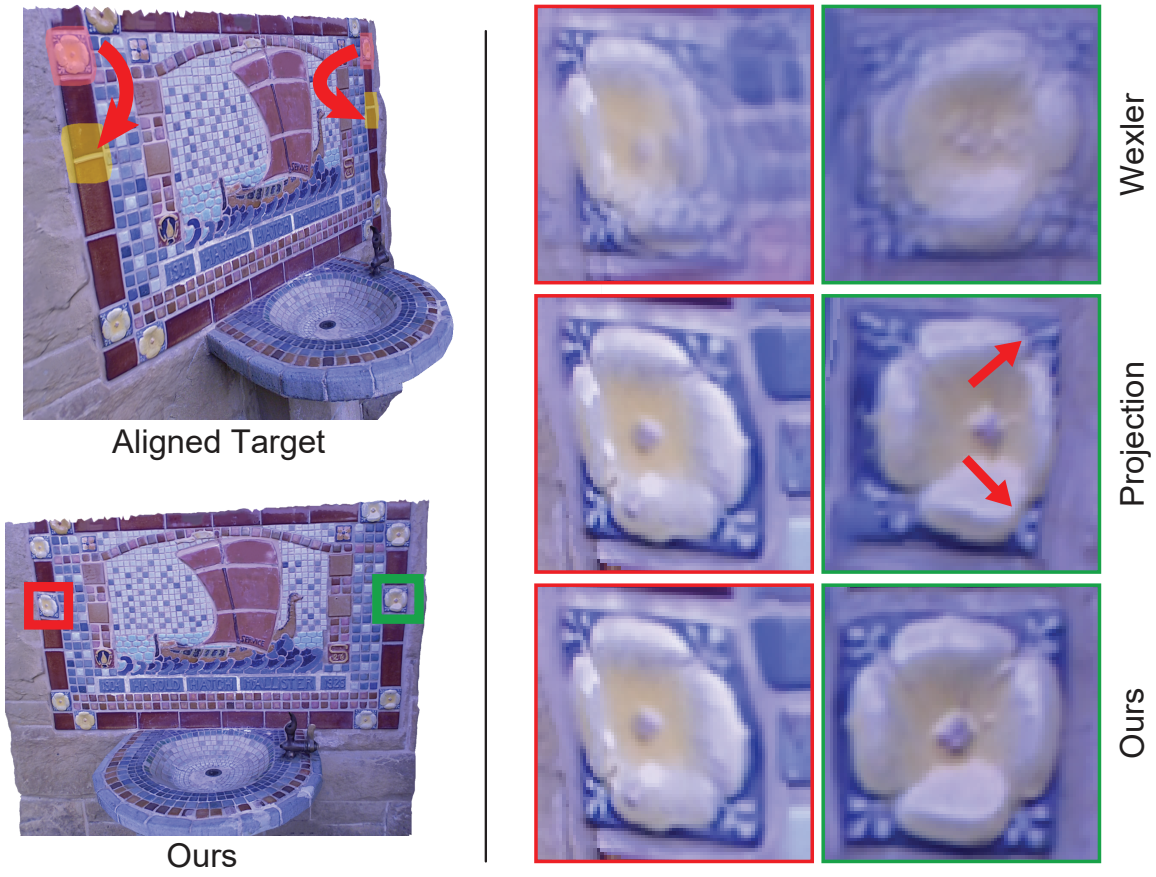

Figure 2.13. We show one of the aligned target images on the left. Here, the goal is to plausibly copy the regions in red to the desired locations which are marked with yellow. We first use Simakov et al. [122] to perform the reshuffling process for this target image. We then project the yellow masks to the other targets and use our hole-filling system to fill in the projected yellow masks. Performing the hole-filling independently for each target image produces inconsistent hole-filled results, which consequently produces textures with ghosting artifacts. Simply projecting the reshuffled result from one target to the other targets has problems at grazing angles. Our method is able to produce consistent results across different views and generate high-quality textures.

Extending our system to also fill in geometries is an interesting topic for future research.

Texture Reshuffling – As shown in Fig. 2.13, our method can also be used to copy parts of a texture (marked with red masks) to other regions within the texture (marked with yellow). Again before starting the reshuffling process we synthesized aligned targets using our system. We then mark some regions in one of the target images (reshuffling target) and the goal is to replicate them in a plausible way in the desired locations (yellow masks in Fig. 2.13). Moreover,

the synthesized contents at the new locations of the reshuffling target need to be consistent with all the other target images.

To do this, we first perform the single image reshuffling [122] and synthesize a replica of the regions of interest in the new locations. Note that, this process is performed exactly like Simakov et al. [122] and only on the reshuffling target. At this point, the other targets are not consistent with this target image in the areas where the reshuffling is performed (yellow regions).

We address this issue, by first projecting the yellow masks to the other targets. We then use our described hole-filling system to fill in the projected yellow masks in other targets. Note that here we do not modify the reshuffling target and it is only used to force the other targets to produce consistent content in the regions defined with the yellow mask. Formally speaking, this means that we remove the E_{BDS} term corresponding to the reshuffling target in Eq. 2.5.

This process produces targets that are consistent with the reshuffling target, as shown in Fig. 2.13. Again, the textures produced by hole-filling each target separately using Wexler et al.'s approach [140] contain ghosting artifacts. Moreover, projecting the content of the yellow mask from the reshuffling target to the other targets produces blurriness. Our method produces high-quality results.

Multiview Camouflage – Our method could also be used to camouflage a 3D object from multiple viewpoints. Here, the input is a set of images of a scene and the geometry of a 3D object that needs to be artificially inserted into the scene and camouflaged. This is done by producing a consistent texture map for the geometry to make it invisible from different viewpoints. This problem can be viewed as image-based texture mapping for a highly inaccurate geometry, where the geometry of the scene is modeled with the 3D object. We compare the result of our technique for camouflaging a box against Owens et al.'s method [103] in Fig. 2.14. Note that, their approach is specifically designed for this application and is limited to camouflaging boxes. Therefore, their approach produces high-quality results in this case. In comparison, our framework is able to handle this additional application and produce reasonable results. Moreover, our method is not limited to boxes and is able to handle any other objects (see supplementary

Figure 2.14. We show three views of a camouflaged box generated by our approach and Owens et al.'s method[103]. Comparing to Owens et al.'s technique, we are able to produce a reasonable texture map.

video of the original publication).

2.7 Conclusion

In this chapter, we have presented a novel global patch-based optimization system for image-based texture mapping. We correct the misalignments caused by the inaccuracies in the geometry, camera poses, and optical distortions of the input images, by synthesizing an aligned image for each source image. We propose to do this using a novel patch-based energy function that reconstructs photometrically-consistent aligned images from the source images. To solve our energy function efficiently, we propose a two step approach involving a modified patch search and vote followed by a reconstruction stage. We show that our patch-based approach is effective in handling large inaccuracies and outperforms state-of-the-art approaches. Moreover, we demonstrate other applications of our system such as texture editing and multiview camouflage.

This chapter is based on the material as it appears in ACM Transactions on Graphics,

2017 (“Patch-Based Optimization for Image-Based Texture Mapping”, Sai Bi, Nima Khademi Kalantari, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 3

Geometry and Reflectance from Sparse Multi-View Images

3.1 Introduction

In Chapter 2, we have introduced a novel method to reproduce high-quality texture maps for 3D reconstructions to reproduce the appearance of real-world objects and scenes. While texture maps are widely used, they are limited to purely diffuse objects. For non-Lambertian objects, we reconstruct their 3D geometry and reflectance properties so that we can render them under novel lighting conditions. Traditionally this has been accomplished using complex acquisition systems [9, 53, 132, 147, 163] or multi-view stereo (MVS) methods [39, 115] applied to dense image sets [97, 152]. The acquisition requirements for these methods significantly limits their practicality. Recently, deep neural networks have been proposed for material estimation from a single or a few images. However, many of these methods are restricted to estimating the spatially-varying BRDF (SVBRDF) of planar samples [33, 42, 82]. Li et al. [83] demonstrate shape and reflectance reconstruction from a single image, but their reconstruction quality is limited by their single image input.

In this chapter, our goal is to enable practical *and* high-quality shape and appearance acquisition. To this end, we propose using a simple capture setup: a sparse set of six cameras—placed at one vertex and the centers of the adjoining faces of a regular icosahedron, forming a 60° cone—with collocated point lighting (Fig. 3.2 left). Capturing six images should allow for

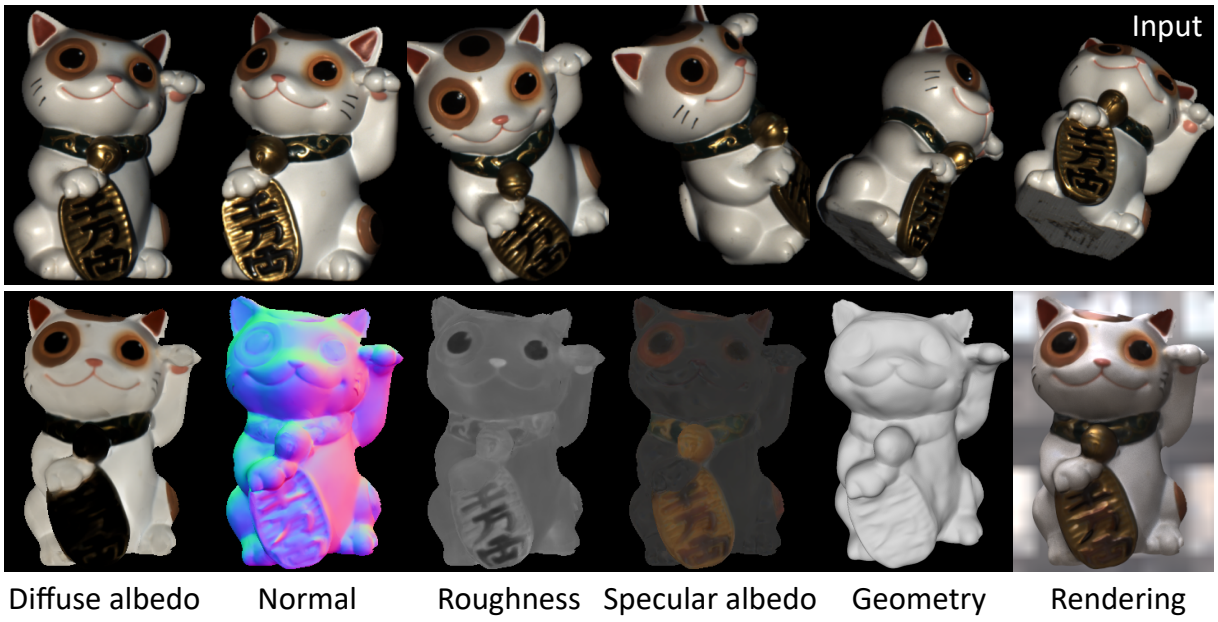

Figure 3.1. From six wide-baseline input images of an object captured under collocated point lighting (top row), our method reconstructs high-quality geometry and spatially-varying, non-Lambertian reflectance (bottom row, a tone mapping is performed on specular albedo to make it more visible), allowing us to re-render the captured object under novel viewpoint and illumination (bottom, right).

better reconstruction compared to single image methods. However, at such wide baselines, the captured images have few correspondences and severe occlusions, making it challenging to fuse information across viewpoints.

As illustrated in Fig. 3.2, we propose a two-stage approach to address this problem. First, we design **multi-view geometry and reflectance estimation networks that regress the 2D depth, normals and reflectance for each input view by robustly aggregating information across all sparse viewpoints.** We estimate the depth for each input view using a deep multi-view stereo network [153, 157] (Sec. 3.3.1). Because of our sparse capture, these depth maps contain errors and cannot be used to accurately align the images to estimate per-vertex BRDFs [97, 163]. Instead, we use these depth maps to warp the images to one viewpoint and use a *novel deep multi-view reflectance estimation network* to estimate per-pixel normals and reflectance (parameterized by diffuse albedo, specular albedo and roughness in a simplified Disney BRDF model [67]) for that viewpoint (Sec. 3.3.2). This network extracts features from the warped images, aggregates

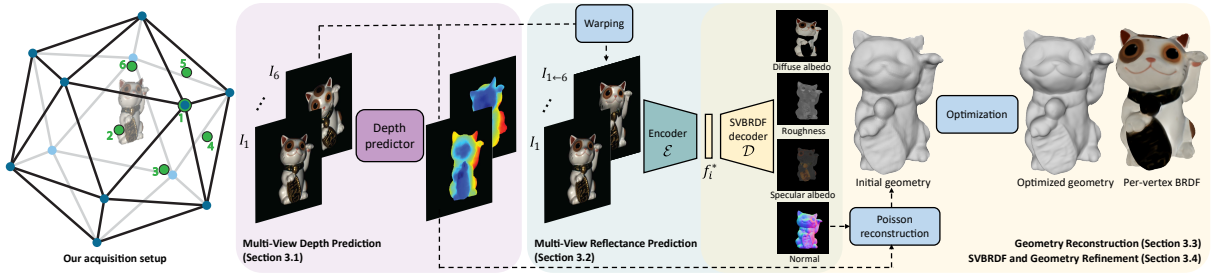

Figure 3.2. Our acquisition setup (leftmost figure) and framework. We capture six images with collocated cameras and lights placed at a vertex (*green* circle 1) and five adjoining face centers (*green* circle 2-6) of an icosahedron. Using the six images, we predict per-view depth (*red block*). We warp the input images using the predicted depths and pass them to a multi-view SVBRDF estimation network to get per-view SVBRDFs (*blue block*). Finally, we reconstruct 3D geometry from the estimated depth and normals, and perform a joint optimization to get refined geometry and per-vertex BRDFs (*yellow block*).

them across viewpoints using max-pooling, and decodes the pooled features to estimate the normals and SVBRDF for that viewpoint. This approach to aggregate multi-view information leads to more robust reconstruction than baseline approaches like a U-Net architecture [112], and we use it to recover normals and reflectance for each view.

Second, we propose a **novel method to fuse these per-view estimates into a single mesh with per-vertex BRDFs using optimization in a learnt reflectance space**. First, we use Poisson reconstruction [68] to construct a mesh from the estimated per-view depth and normal maps (Sec. 3.3.3). Each mesh vertex has multiple reflectance parameters corresponding to each per-view reflectance map, and we fuse these estimates to reconstruct object geometry and reflectance that will *accurately reproduce the input images*. Instead of optimizing the per-vertex reflectance parameters, which leads to outliers and spatial discontinuities, we optimize the *the latent features of our multi-view reflectance estimation network* (Sec. 3.3.4). We pass these latent features to the reflectance decoder to construct per-view SVBRDFs, fuse them using per-vertex blending weights, and render them to compute the photometric error for all views. This entire pipeline is differentiable, allowing us to backpropagate this error and iteratively update the reflectance latent features and per-vertex weights till convergence. This process refines the reconstruction to best match the specific captured images, while leveraging the priors learnt by

our reflectance estimation network.

We train our networks with a large-scale synthetic dataset comprised of procedurally generated shapes with complex SVBRDFs [153, 155] and rendered using a physically-based renderer. While our method is trained with purely synthetic data, it generalizes well to real scenes. This is illustrated in Figs. 3.1 and 3.8, where we are able to reconstruct real objects with complex geometry and non-Lambertian reflectance. Previous state-of-the-art methods, when applied to sparse input images for such objects, produce incomplete, noisy geometry and erroneous reflectance estimates (Figs. 3.4 and 3.7). In contrast, our work is the first to reconstruct detailed geometry and high-quality reflectance from sparse multi-view inputs, allowing us to render photorealistic images under novel view and lighting.

3.2 Related Works

3D reconstruction. To reconstruct 3D geometry from image sets, traditional methods [40, 76, 115] find correspondences between two or more images utilizing specific image features. Such methods are sensitive to illumination changes, non-Lambertian reflectance and textureless surfaces. The existence of multiple points with similar matching costs also require these methods to have a large number of images to get high-quality reconstructions (we refer the interested readers to [40] for more details). In contrast, our method reconstructs high-quality geometry for complex real scenes from an order of magnitude fewer images.

Recently, numerous learning-based methods have been proposed to reconstruct 3D shape using various geometric representations, including regular volumes [60, 110, 149], point clouds [1, 136] and depth maps [54, 157]. These methods cannot produce high-resolution 3D meshes. We extend recent learning-based MVS frameworks [153, 157] to estimate depth from sparse multi-view images of objects with complex reflectance. We combine this depth with estimated surface normals to reconstruct 3D meshes with fine details.

SVBRDF acquisition. SVBRDF acquisition is a challenging task that often requires a

dense input image set [35, 97, 152]. Many methods utilize sophisticated hardware [89] or light patterns [53, 64, 132]. Reconstruction from sparse images has been demonstrated for planar objects [5, 82, 154], and known geometry [163]. In contrast, we reconstruct the geometry and complex reflectance of arbitrary objects from a sparse set of six input images.

Photometric stereo methods have been proposed to reconstruct arbitrary shape and SVBRDFs [7, 45]; however, they focus on single-view reconstruction and require hundreds of images. Recent works [55, 97] utilize images captured by a collocated camera-light setup for shape and SVBRDF estimation. In particular, Nam et al. [97] capture more than sixty images and use multi-view reconstruction and physics-based optimization to recover geometry and reflectance. In contrast, by designing novel deep networks, we are able to reconstruct objects from only six images.

Learning-based methods have been applied for normal and SVBRDF acquisition. Deep photometric stereo methods reconstruct surface normals from tens to hundreds of images [21, 22] but they do not address reflectance or 3D geometry estimation. Most deep SVBRDF acquisition methods are designed for planar samples [4, 33, 34, 42, 81, 82]. Some recent multi-image SVBRDF estimation approaches pool latent features from multiple views [34] and use latent feature optimization [42] but they only handle planar objects. Li et al. [83] predict depth and SVBRDF from a single image; however, a single input does not provide enough information to accurately reconstruct geometry and reflectance. By capturing just six images, our approach generates significantly higher quality results.

3.3 Algorithm

Our goal is to accurately reconstruct the geometry and SVBRDF of an object with a simple acquisition setup. Recent work has utilized collocated point illumination for reflectance estimation from a sparse set of images [4, 5, 33, 82]; such lighting minimizes shadows and induces high-frequency effects like specularities, making reflectance estimation easier. Similarly,

Xu et al. [153] demonstrate novel view synthesis from sparse multi-view images of a scene captured under a single point light.

Motivated by this, we utilize a similar capture system as Xu et al.—six cameras placed at one vertex of a regular icosahedron and the centers of the five faces adjoining that vertex. Unlike their use of a single point light for all images, we capture each image under a point light (nearly) collocated with the corresponding camera (see Fig. 3.2 left). The setup is calibrated giving us a set of $n = 6$ input images, $\{I_i\}_{i=1}^{n}$ with the corresponding camera calibration. This wide baseline setup—with an angle of 37° between the center and boundary views—makes it possible to image the entire object with a small set of cameras. In the following, we describe how we reconstruct an object from these sparse input images.

3.3.1 Multi-View Depth Prediction

Traditional MVS methods depend on hand-crafted features such as Harris descriptors to find correspondence between views. Such features are not robust to illumination changes or non-Lambertian surfaces, making them unusable for our purposes. In addition, due to the sparse inputs and large baselines, parts of the object may be visible in as few as two views. These factors cause traditional MVS methods to fail at finding accurate correspondences, and thus fail to reconstruct high-quality geometry.

Instead, we make use of a learning-based method to estimate the depth. Given the input images $\{I_i\}_{i=1}^{n}$, we estimate the depth map D_i for view i. Similar to recent works on learning-based MVS [56, 153, 157], our network consists of two components: a feature extractor $\mathscr{F}$ and a correspondence predictor $\mathscr{C}$. The feature extractor is a 2D U-Net [112] that extracts a 16-channel feature map for each image I_i. To estimate the depth map at I_i, we warp the feature maps of all views to view i using a set of 128 pre-defined depth levels, and build a 3D plane sweep volume [26] by calculating the variance of feature maps over views. The 3D volume is further fed to the correspondence predictor $\mathscr{C}$ that is a 3D U-Net to predict the probability of each depth level. We calculate the depth as a probability-weighted sum of all depth levels.

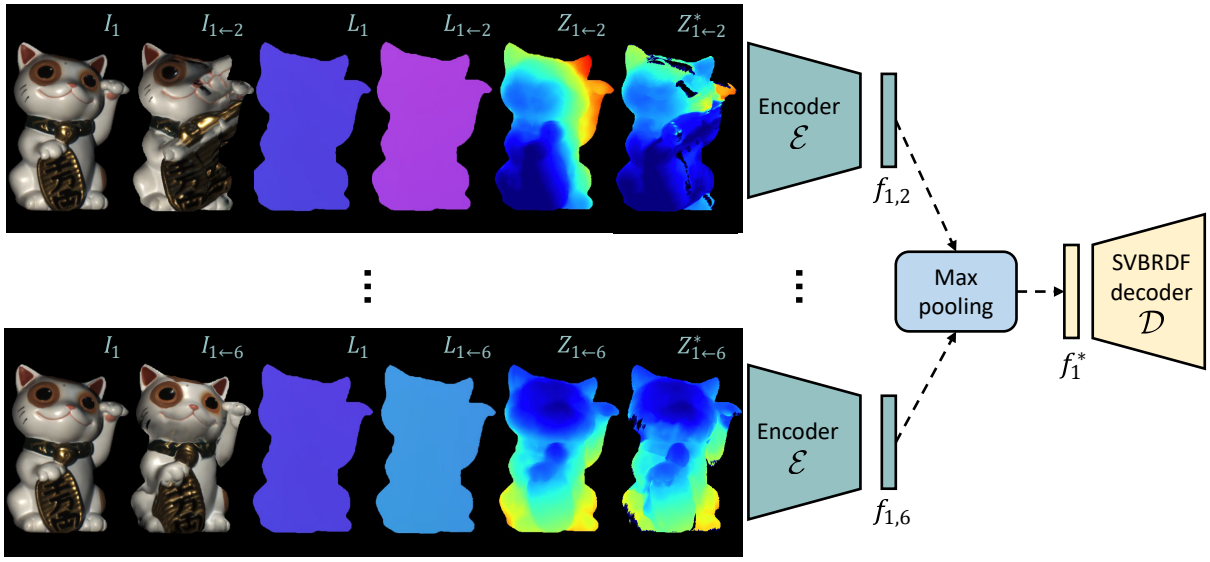

Figure 3.3. Our multi-view SVBRDF estimation network. An encoder extracts features from reference and warped image pairs. These features are max-pooled to get a single reference-view feature map, which is decoded to predict that view's SVBRDF. Note the errors in the warped images; max-pooling mitigates their effect on the output SVBRDF.

The training loss is defined as the L_1 loss between predicted depths and ground truth depths. By learning the feature representations and correspondence, the proposed framework is more robust to illumination changes and specularities, thus producing more accurate pixel-wise depth predictions than traditional methods.

While such networks are able to produce reasonable depth, the recovered depth has errors in textureless regions. To further improve the accuracy, we add a guided filter module [148] to the network, which includes a guided map extractor $\mathscr{G}$ as well as a guided layer g. Let the initial depth prediction at view i be D'_i. The guided map extractor $\mathscr{G}$ takes image I_i as input and learns a guidance map $\mathscr{G}(I_i)$. The final depth map is estimated as:

$$D_i = g(\mathscr{G}(I_i), D'_i). \tag{3.1}$$

The training loss is defined as the L_1 distance between predicted depths and ground truth depths. All components are trained jointly in an end-to-end manner.

3.3.2 Multi-View Reflectance Prediction

Estimating surface reflectance from sparse images is a highly under-constrained problem. Previous methods either assume geometry is known [4, 5, 82, 33] or can be reconstructed with specific devices [53] or MVS [97]. In our case, accurate geometry cannot be reconstructed from sparse inputs with traditional MVS methods. While our learning-based MVS method produces reasonable depth maps, they too have errors, making it challenging to use them to align the images and estimate per-vertex SVBRDF. Instead, for each input image I_i, we first estimate its corresponding normals, I_i, and SVBRDF, represented by diffuse albedo A_i, specular roughness R_i and specular albedo S_i.

To estimate the SVBRDF at view i, we warp all input images to this view using predicted depths D_i. One approach for multi-view SVBRDF estimation could be to feed this stack of warped images to a convolutional neural network like the commonly used U-Net [82, 112]. However, the inaccuracies in the depth maps lead to misalignments in the warped images, especially in occluded regions, and this architecture is not robust to these issues.

We propose a novel architecture that is robust to depth inaccuracies and occlusions. As shown in Fig. 3.3, our network comprises a Siamese encoder [25], $\mathscr{E}$, and a decoder, $\mathscr{D}$, with four branches for the four SVBRDF components. To estimate the SVBRDF at a reference view i, the encoder processes n pairs of inputs, each pair including image I_i as well as the warped image $I_{i\leftarrow j}$, where we warp image I_j at view j to the reference view i using the predicted depth D_i. To handle potential occlusions, directly locating occluded regions in the warped images using predicted depths and masking them out is often not feasible due to inaccurate depths. Instead we keep the occluded regions in the warped images and include the depth information in the inputs, allowing the network to learn which parts are occluded.

To include the depth information, we draw inspiration from the commonly used shadow mapping technique [142]. The depth input consists of two components: for each pixel in view i, we calculate its depths $Z_{i\leftarrow j}$ in view j; we also sample its depth $Z^*_{i\leftarrow j}$ from the depth map

D_j by finding its projections on view j. Intuitively if $Z_{i\leftarrow j}$ is larger than $Z^*_{i\leftarrow j}$, then the pixel is occluded in view j; otherwise it is not occluded. In addition, for each pixel in the reference view i, we also include the lighting directions L_i of the light at view i, as well as the lighting direction of the light at view j, denoted as $L_{i\leftarrow j}$. We assume a *point light* model here. Since the light is collocated with the camera, by including the lighting direction we are also including the viewing direction of each pixel in the inputs. All directions are in the coordinate system of the reference view. Such cues are critical for networks to infer surface normals using photometric information. Therefore, the input for a pair of views i and j is:

$$H_{i,j} = \{I_i, I_{i\leftarrow j}, Z_{i\leftarrow j}, Z^*_{i\leftarrow j}, L_i, L_{i\leftarrow j}\}. \tag{3.2}$$

The input contains 14 channels in total, and there are a total of n such inputs. We feed all the inputs to the encoder network $\mathscr{E}$ and get the intermediate features $f_{i,j}$. All these intermediate features are aggregated with a max-pooling layer yielding a common feature representation for view i, f^*_i:

$$f_{i,j} = \mathscr{E}(H_{i,j}) \tag{3.3}$$

$$f^*_i = \text{max-pool}(\{f_{i,j}\}^n_{j=1}) \tag{3.4}$$

f^*_i is fed to the decoder to predict each SVBRDF component for view i:

$$A_i, N_i, R_i, S_i = \mathscr{D}(f^*_i) \tag{3.5}$$

Compared to directly stacking all warped images together, our proposed network architecture works on pairs of input images and aggregates features across views using a max-pooling layer. The use of max-pooling makes the network more robust to occlusions and misalignments caused by depth inaccuracies and produces more accurate results (see Tab. B.1). It also makes the

network invariant to the number and order of the input views, a fact that could be utilized for unstructured capture setups. The training loss $\mathscr{L}$ of the network is defined as:

$$\mathscr{L} = \mathscr{L}_A + \mathscr{L}_N + \mathscr{L}_R + \mathscr{L}_S + \mathscr{L}_I \tag{3.6}$$

where the first four terms are the L_2 losses for each SVBRDF component, and $\mathscr{L}_I$ is the L_2 loss between input images and rendered images generated with our predictions.

3.3.3 Geometry Reconstruction

The previous multi-view depth and SVBRDF estimation networks give us per-view depth and normal maps at full-pixel resolution. We fuse these per-view estimates to reconstruct a single 3D geometry for the object. We first build a point cloud from the depth maps, by generating 3D points from each pixel in every per-view depth map. For each point, we also get its corresponding normal from the estimated normal maps. Given this set of 3D points with surface normals, we perform a Poisson reconstruction [69] to reconstruct the fused 3D geometry. The initial point clouds may contain outliers due to inaccuracies in the depth maps. To get rid of undesired structures in the output geometry, we generate a coarse initial geometry by setting the depth of the spatial octree in Poisson reconstruction to 7—corresponding to an effective voxel resolution of 128^3. We refine this initial geometry in the subsequent stage. Compared to learning-based 3D reconstruction methods that directly generate geometry (voxel grids [66, 111], implicit functions [104, 113] or triangle meshes [137]) from images, this approach generalizes to arbitrary shapes and produces more detailed reconstructions.

3.3.4 SVBRDF and Geometry Refinement

Given the initial coarse geometry as well as the per-view SVBRDF predictions, we aim to construct a detailed 3D mesh with per-vertex BRDFs. For each vertex, a trivial way to get its BRDF is to blend the predicted SVBRDFs across views using pre-defined weights such as the

dot product of the viewing directions and surface normals. However, this leads to blurry results (Fig. 3.5), due to the inconsistencies in the estimated SVBRDFs and the geometry. Also note that our SVBRDF predictions are computed from a single feed-forward network pass, and are not guaranteed to reproduce the captured input images exactly because the network has been trained to minimize the reconstruction loss on the entire training set and not this specific input sample.

We address these two issues with a novel rendering-based optimization that estimates per-vertex BRDFs that minimize the error between rendering the predicted parameters and the captured images. Because of the sparse observations, independently optimizing per-vertex BRDFs leads to artifacts such as outliers and spatial discontinuities, as shown in Fig. 3.5. Classic inverse rendering methods address this using hand-crafted priors. Instead, we optimize the per-view feature maps f_i^* that are initially predicted from our SVBRDF encoder (Eqn. 3.4). These latent features, by virtue of the training process, capture the manifold of object reflectances, and generate spatially coherent per-view SVBRDFs when passed through the decoder, $\mathscr{D}$ (Eqn. 3.5). *Optimizing in this feature space allows us to adapt the reconstruction to the input images, while leveraging the priors learnt by our multi-view SVBRDF estimation network.*

Per-vertex BRDF and color. For each vertex v_k, we represent its BRDF b_k as a weighted average of the BRDF predictions from multiple views:

$$b_k = \sum_{i=1}^{n} w_{k,i} \mathscr{D}(p_{k,i}; f_i^*), \tag{3.7}$$

where $p_{k,i}$ is the corresponding pixel position of v_k at view i, $\mathscr{D}(p_{k,i}; f_i^*)$ represents the SVBRDF prediction at $p_{k,i}$ from view i by processing f_i^* via the decoder network $\mathscr{D}$, and $w_{k,i}$ are the per-vertex view blending weights. The rendered color of v_k at view i is calculated as:

$$I_i^*(p_{k,i}) = \Theta(b_k, L_i(p_{k,i})), \tag{3.8}$$

where $L_i(p_{k,i})$ is the lighting direction and also the viewing direction of vertex v_k at view i, and

Θ is the rendering equation. We assume a point light source collocated with the camera (which allows us to ignore shadows), and only consider direct illumination in the rendering equation.

Per-view warping. Vertex v_k can be projected onto view i using the camera calibration; we refer to this projection as $u_{k,i}$. However, the pixel projections onto multiple views might be inconsistent due to inaccuracies in the reconstructed geometry. Inspired by Zhou et al. [161], we apply a non-rigid warping to each view to better align the projections. In particular, for each input view, we use a $T \times T$ grid with $C = T^2$ control points ($T = 11$ in our experiments) to construct a smooth warping field over the image plane. Let $t_{i,c}$ be the translation vectors of control points at view i. The resulting pixel projection, $p_{k,i}$, is given by:

$$p_{k,i} = u_{k,i} + \sum_{c=1}^{C} \theta_c(u_{k,i}) t_{i,c}, \tag{3.9}$$

where θ_c returns the bilinear weight for a control point $t_{i,c}$ at pixel location $u_{k,i}$.

SVBRDF optimization. We optimize per-view latent features f_i^*, per-vertex blending weights $w_{k,i}$ and per-view warping fields $t_{i,c}$ to reconstruct the final SVBRDFs. The photometric consistency loss between the rendered colors and ground truth colors for all K vertices is given by:

$$E_{\text{photo}}(f_i^*, w, t) = \frac{1}{n \cdot K} \sum_{k=1}^{K} \sum_{i=1}^{n} ||I_i^*(p_{k,i}) - I_i(p_{k,i})||_2^2.$$

We clamp the rendered colors to the range of $[0,1]$ before calculating the loss. To prevent the non-rigid warping from drifting, we also add an L_2 regularizer to penalize the norm of the translation vectors:

$$E_{\text{warp}}(t) = \frac{1}{n \cdot C} \sum_{i=1}^{n} \sum_{c=1}^{C} ||t_{i,c}||_2^2. \tag{3.10}$$

Therefore the final energy function for the optimization is:

$$E = E_{\text{photo}}(f^*, w, t) + \lambda E_{\text{warp}}(t). \tag{3.11}$$

We set λ to 100 , and optimize the energy function with Adam optimizer [70] with a learning rate of 0.001.

Geometry optimization. We use the optimized per-vertex normal, n_k, to update the geometry of the object by re-solving the Poisson equation (Sec. 3.3.3). Unlike the initial geometry reconstruction, we set the depth of the spatial octree to 9—corresponding to a voxel resolution of 512^3—to better capture fine-grained details of the object. We use this updated geometry in subsequent SVBRDF optimization iterations. We update the geometry once for every 50 iterations of SVBRDF optimization, and we perform $400-1000$ iterations for the SVBRDF optimization.

Per-vertex refinement. The bottleneck in our multi-view SVBRDF network—that we use as our reflectance representation—may cause a loss of high-frequency details in the predicted SVBRDFs. We retrieve these details back by directly optimizing the BRDF parameters b_k of each vertex to minimizing the photometric loss in Eqn. (3.10). Note that after the previous optimization, the estimated BRDFs have already converged to good results and the rendered images are very close to the input images. Therefore, in this stage, we use a small learning rate (0.0005), and perform the optimization for a small number $(40-100)$ of iterations.

3.4 Implementation and Results

Training data. We follow Xu et al. [153] and procedurally generate complex scenes by combining 1 to 5 primitive shapes such as cylinders and cubes displaced by random height maps. We generate $20,000$ training and 400 testing scenes. We divide the high-quality materials from

the Adobe Stock dataset[1] into a training and testing set, and use them to texture the generated scenes separately. For each scene, following the setup discussed in Sec. 5.1, we render the 6 input view images with a resolution of 512×512 using a custom Optix-based global illumination renderer with 1000 samples per pixel. We also render the ground truth depth, normals, and SVBRDF components for each view.

Network architecture. For depth estimation, we use a 2D U-Net architecture [112] for the feature extractor, $\mathscr{F}$, and guidance map extractor, $\mathscr{G}$. Both networks have 2 downsampling/upsampling blocks. The correspondence predictor $\mathscr{C}$ is a 3D U-Net with 4 downsampling/upsampling blocks. For multi-view SVBRDF estimation, both the encoder $\mathscr{E}$ and decoder $\mathscr{D}$ are 2D CNNs, with 3 downsampling layers in $\mathscr{E}$ and 3 upsampling layers in $\mathscr{D}$. Note that we *do not* use skip connections in the SVBRDF network; this forces the latent feature to learn a meaningful reflectance space and allows us to optimize it in our refinement step. We use group normalization [150] in all networks. We use a differentiable rendering layer that computes local shading under point lighting without considering visibility or global illumination. This is a reasonable approximation in our collocated lighting setup. For more details, please refer to the supplementary document.

Training details. All the networks are trained with the Adam optimizer [70] for 50 epochs with a learning rate of 0.0002. The depth estimation networks are trained on cropped patches of 64×64 with a batch size of 12, and the SVBRDF estimation networks are trained on cropped 320×320 patches with a batch size of 8. Training took around four days on 4 NVIDIA Titan 2080Ti GPUs.

Run-time. Our implementaion has not been optimized for the best timing efficiency. In practice, our method takes around 15 minutes for full reconstruction from images with a resolution of 512×512, where most of the time is for geometry fusion and optimization.

[1] https://stock.adobe.com/search/3d-assets

Table 3.1. Quantitative SVBRDF evaluation on a synthetic test set. We report the L_2 error. Since Li et al. [83] work on 256×256 images, we downsample and evaluate at that resolution. Also, they do not predict the specular albedo.

	Diffuse	Normal	Roughness	Specular
Naive U-Net	**0.0060**	0.0336	0.0359	0.0125
Ours	0.0061	**0.0304**	**0.0275**	**0.0086**
Li et al. [83]	0.0227	0.1075	0.0661	—
Ours (256×256)	**0.0047**	**0.0226**	**0.0257**	**0.0083**

3.4.1 Evaluation on Synthetic Data

We evaluate our max-pooling-based multi-view SVBRDF estimation network on our synthetic test set. In particular, we compare it with a baseline U-Net (with 5 downsampling/upsampling blocks) that takes a stack of all the coarsely aligned images ($H_{i,j} \forall j$ in Eqn. 3.2) as input for its encoder, and skip connections from the encoder to the four SVBRDF decoders. This architecture has been widely used for SVBRDF estimation [33, 82, 83]. As can be seen in Tab. B.1, while our diffuse albedo prediction is slightly (1.7%) worse than the U-Net we significantly outperform it in specular albedo, roughness and normal predictions, with 31%, 23% and 9.5% lower L_2 loss respectively. This is in spite of not using skip-connections in our network (to allow for optimization later in our pipeline). We also compare our results with the state-of-the-art single-image shape and SVBRDF estimation method of Li et al. [83]. Unsurprisingly, we outperform them significantly, demonstrating the usefulness of aggregating multi-view information.

3.4.2 Evaluation on Real Captured Data

We evaluate our method on real data captured using a gantry with a FLIR camera and a nearly collocated light to mimic our capture setup. **Please refer to the supplementary material for additional results.**

Evaluation of geometry reconstruction. Our framework combines our predicted depths

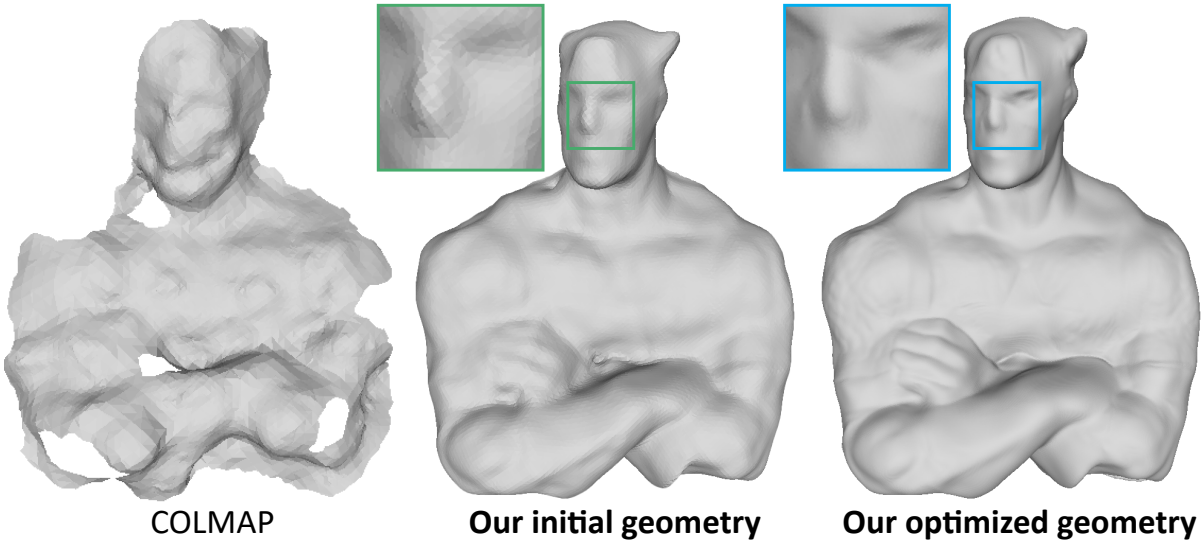

Figure 3.4. Comparison on geometry reconstruction. COLMAP fails to reconstruct a complete mesh from the sparse inputs. In contrast, our initial mesh is of much higher quality, and our joint optimization recovers even more fine-grained details on the mesh. Input image in Fig. 3.8 (top).

and normals to reconstruct the initial mesh. Figure 3.4 shows the comparison between our reconstructed mesh and the mesh from COLMAP, a state-of-the-art multi-view stereo framework [115]. From such sparse inputs and low-texture surfaces, COLMAP is not able to find reliable correspondence across views, which results in a noisy, incomplete 3D mesh. In contrast, our initial mesh is already more complete and detailed, as a result of our more accurate depths and normals. Our joint optimization further refines the per-vertex normals and extracts fine-scale detail in the object geometry.

Evaluation of SVBRDF optimization. We compare our SVBRDF and geometry optimization scheme (Sec. 3.3.4) with averaging the per-view predictions using weights based on the angle between the viewpoint and surface normal, as well as this averaging followed by per-vertex optimization. From Fig. 3.5 we can see that the weighted averaging produces blurry results. Optimizing the per-vertex BRDFs brings back detail but also has spurious discontinuities in appearance because of the lack of any regularization. In contrast, our latent-space optimization method recovers detailed appearance without these artifacts.

Figure 3.5. Comparison on SVBRDF optimization. Simple averaging without optimization produces blurry results, and direct per-vertex optimization results in outliers and discontinuities. In comparison, our optimization generates more visually plausible results.

Comparisons against Nam et al. [97] We also compare our work with the state-of-the-art geometry and reflectance reconstruction method of Nam et al. Their work captures 60+ images of an object with a handheld camera under collocated lighting; they first use COLMAP [115] to reconstruct the coarse shape and use it to bootstrap a physics-based optimization process to recover per-vertex normals and BRDFs. COLMAP cannot generate complete meshes from our sparse inputs (see Fig. 3.4). Therefore, we provided our input images, camera calibration, *and initial geometry* to the authors who processed this data. As can be seen in Fig. 3.6, our final reconstructed geometry has significantly more details than their final optimized result in spite of starting from the same initialization. Since they use a different BRDF representation than ours, making direct SVBRDF comparisons difficult, in Fig. 3.7 we compare renderings of the reconstructed object under novel lighting and viewpoint. These results show that they cannot handle our sparse input and produce noise, erroneous reflectance (CAT scene) or are unable to recover the specular highlights of highly specular objects (CACTUS) scene. In comparison, our

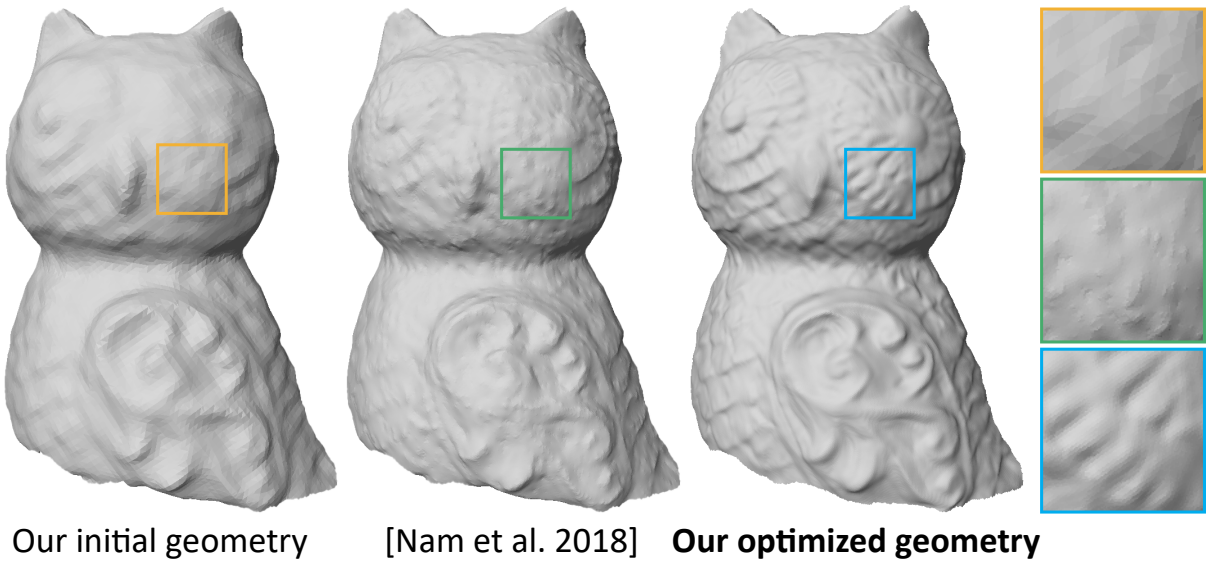

Figure 3.6. Comparison with Nam et al. [97]. While both have the same initialization, our learning-based refinement produces more accurate, detailed geometry. Input in Fig. 3.8.

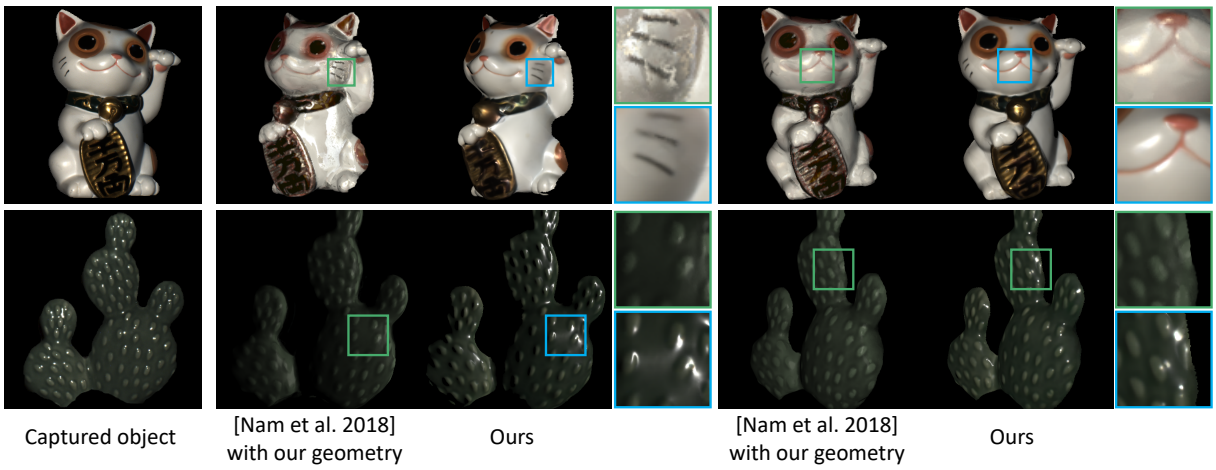

Figure 3.7. Comparison with Nam et al. [97]. We render two reconstructed objects under *novel* viewpoints and lighting. Nam et al. are not able to accurately reconstruct appearance from sparse views, and produce noisy edges and incorrect specular highlights (top) or miss the specular component completely (bottom). In contrast, our method produces photorealistic results.

results have significantly higher visual fidelity. Please refer to the supplementary video of the original publication for more renderings.

More results on real data. Figure 3.8 shows results from our method on additional real scenes. We can see here that our method can reconstruct detailed geometry and appearance for

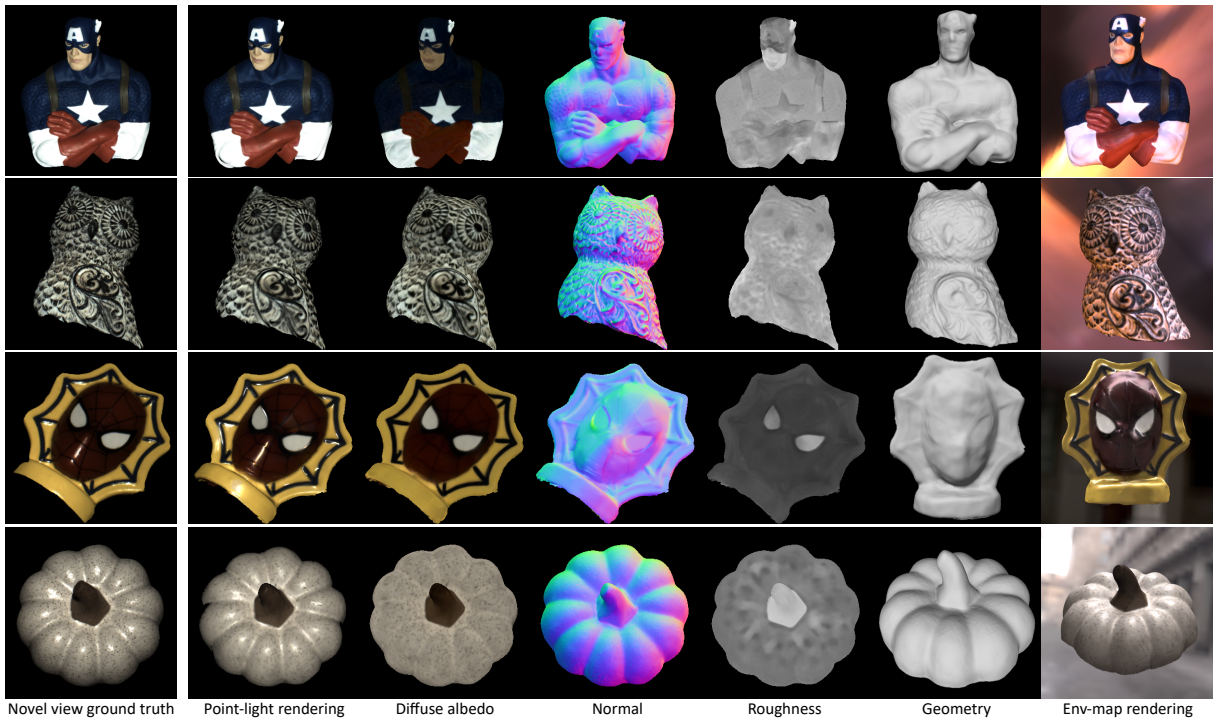

Figure 3.8. Results on real scenes. For each scene, we show our reconstructed geometry, normal map and SVBRDF components (please refer to supplementary materials for specular albedo). We compare our point-light rendering results (second column) under *novel* viewpoints and lighting with captured ground truth photographs (first column). We also show a rendering of the object with our reconstructed appearance under environment lighting (last column).

objects with a wide variety of complex shapes and reflectance. Comparing renderings of our estimates under novel camera and collocated lighting against ground truth captured photographs demonstrates the accuracy of our reconstructions. We can also photorealistically render these objects under novel environment illumnination. Please refer to the supplementary document and video of the original publication for more results.

Limitations. Our method might fail to handle highly non-convex objects, where some parts are visible in as few as a single view and there are no correspondence cues to infer correct depth. In addition, we do not consider global illumination in SVBRDF optimization. While it is a reasonable approximation in most cases, it might fail in some particular scenes with strong inter-reflections. For future work, it would be interesting to combine our method with physics-based differentiable rendering [80, 158] to handle these complex light transport effects.

3.5 Conclusion

We have proposed a learning-based framework to reconstruct the geometry and appearance of an arbitrary object from a sparse set of just six images. We predict per-view depth using learning-based MVS, and design a novel multi-view reflectance estimation network that robustly aggregates information from our sparse views for accurate normal and SVBRDF estimation. We further propose a novel joint optimization in latent feature space to fuse and refine our multi-view predictions. Unlike previous methods that require densely sampled images, our method produces high-quality reconstructions from a sparse set of images, and presents a step towards practical appearance capture for 3D scanning and VR/AR applications.

This chapter is based on the material as it appears in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019 ("Deep 3D Capture: Geometry and Reflectance from Sparse Multi-View Images", Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 4

Deep Reflectance Volumes for Relightable Appearance Acquisition

4.1 Introduction

In the first two chapters, we reconstruct the scene geometry in the form of triangle meshes. However, it remains highly challenging to accurately reconstruct mesh-based geometry for complex scenes such as those with thin structures, textureless regions, heavy occlusions, and non-convex shapes. Moreover, triangle meshes have irregular structures, which makes it difficult to be integrated with neural networks for effective training. In contrast, volumes have the regular 3D grids and are widely used to represent scenes with complex geometry.

Therefore, in this chapter, we propose a novel volumetric representation to make high-quality scene acquisition and rendering practical with off-the-shelf devices under mildly controlled conditions. We use a set of unstructured images captured around a scene by a single mobile phone camera with flash illumination in a dark room. This practical setup acquires multi-view images under collocated viewing and lighting directions—referred to as photometric images [153]. While the high-frequency appearance variation in these images (due to sharp specular highlights and shadows) can result in low-quality mesh reconstruction from state-of-the-art methods (see Fig. 4.3), we show that our method can accurately model the scene and realistically reproduce complex appearance information like specularities and occlusions.

At the heart of our method is a novel, physically-based neural volume rendering frame-

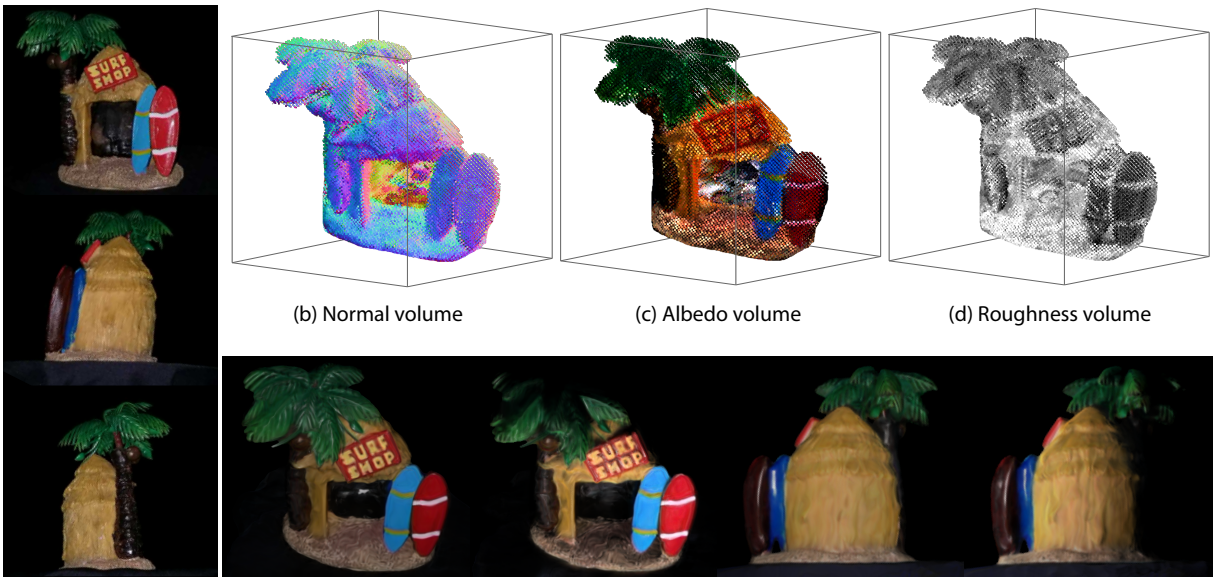

Figure 4.1. Given a set of images taken using a mobile phone with flashlight (sampled images are shown in (a)), our method learns a volume representation of the captured object by estimating the opacity volume, normal volume (b) and reflectance volumes such as albedo (c) and roughness (d). Our volume representation enables free navigation of the object under arbitrary viewpoints and novel lighting conditions (e).

work. We train a deep neural network that simultaneously learns the geometry and *reflectance* of a scene as volumes. We leverage a decoder-like network architecture, where an encoding vector together with the corresponding network parameters are learned during a per-scene optimization (training) process. Our network decodes a volumetric scene representation consisting of opacity, normal, diffuse color and roughness volumes, which model the global geometry, local surface orientations and spatially-varying reflectance parameters of the scene, respectively. These volumes are supplied to a differentiable rendering module to render images with collocated light-view settings at training time, and arbitrary light-view settings at inference time (see Fig. 4.2).

We base our differentiable rendering module on classical volume ray marching approaches with opacity (alpha) accumulation and compositing [73, 144]. In particular, we compute point-wise shading using local normal and reflectance properties, and accumulate the shaded colors with opacities along each marching ray of sight. Unlike the opacity used in previous view synthesis work [86, 162] that is only accumulated along view directions, we propose to

learn global scene opacity that can be accumulated from both view and light directions. As shown in Fig. 4.1, we demonstrate that our scene opacity can be effectively learned and used to compute accurate hard shadows under *novel lighting*, despite the fact that the training process never observed images with shadows that are taken under non-collocated view-light setups. Moreover, different from previous volume-based works [86, 162] that learn a single color at each voxel, we reconstruct per-voxel reflectance and handle complex materials with high glossiness. Our neural rendering framework thus enables rendering with complex view-dependent and light-dependent shading effects including specularities, occlusions and shadows. We compare against a state-of-the-art mesh-based method [97], and demonstrate that our method is able to achieve more accurate reconstructions and renderings (see Fig. 4.3). We also show that our approach supports scene material editing by modifying the reconstructed reflectance volumes (see Fig. 4.6). To summarize, our contributions are:

- A practical neural rendering framework that reproduces high-quality geometry and appearance from unstructured mobile phone flash images and enables view synthesis, relighting, and scene editing.
- A novel scene appearance representation using opacity, normal and reflectance volumes.
- A physically-based differentiable volume rendering approach based on deep priors that can effectively reconstruct the volumes from input flash images.

4.2 Related Works

Geometry reconstruction. There is a long history in reconstructing 3D geometry from images using traditional structure from motion and multi-view stereo (MVS) pipelines [40, 74, 115]. Recently deep learning techniques have also been applied to 3D reconstruction with various representations, including volumes [60, 110], point clouds [1, 105, 136], depth maps [54, 157] and implicit functions [24, 93, 100]. We aim to model scene geometry for realistic image synthesis, for which mesh-based reconstruction [68, 87, 98] is the most common way in

many applications [14, 97, 107, 161]. However, it remains challenging to reconstruct accurate meshes for challenging scenes where there are textureless regions and thin structures, and it is hard to incorporate a mesh into a deep learning framework [75, 84]; the few mesh-based deep learning works [48, 63] are limited to category-specific reconstruction and cannot produce photo-realistic results. Instead, we leverage a physically-based opacity volume representation that can be easily embedded in a deep learning system to express scene geometry of arbitrary shapes.

Reflectance acquisition. Reflectance of real materials is classically measured using sophisticated devices to densely acquire light-view samples [37, 89], which is impractical for common users. Recent works have improved the practicality with fewer samples [99, 154] and more practical devices (mobile phones) [4, 5, 55, 82]; however, most of them focus on flat planar objects. A few single-view techniques based on photometric stereo [7, 45] or deep learning [83] are able to handle arbitrary shape, but they merely recover limited single-view scene content. To recover complete shape with spatially varying BRDF from multi-view inputs, previous works usually rely on a pre-reconstructed initial mesh and images captured under complex controlled setups to reconstruct per-vertex BRDFs [16, 65, 147, 152, 163]. While a recent work [97] uses a mobile phone for practical acquisition like ours, it still requires MVS-based mesh reconstruction, which is ineffective for challenging scenes with textureless, specular and thin-structure regions. In contrast, we reconstruct spatially varying volumetric reflectance via deep network based optimization; we avoid using any initial geometry and propose to jointly reconstruct geometry and reflectance in a holistic framework.

Relighting and view synthesis. Image-based techniques have been extensively explored in graphics and vision to synthesize images under novel lighting and viewpoint without explicit complete reconstruction [17, 30, 79, 106]. Recently, deep learning has been applied to view synthesis and most methods leverage either view-dependent volumes [125, 153, 162] or canonical world-space volumes [86, 124] for geometric-aware appearance inference. We extend them

to a more general physically-based volumetric representation which explicitly expresses both geometry and reflectance, and enables relighting with view synthesis. On the other hand, learning-based relighting techniques have also been developed. Purely image-based methods are able to relight scenes with realistic specularities and soft shadows from sparse inputs, but unable to reproduce accurate hard shadows [62, 128, 155, 160]; some other methods [23, 107] propose geometry-aware networks and make use of pre-acquired meshes for relighting and view synthesis, and their performance is limited by the mesh reconstruction quality. A work [95] concurrent to ours models scene geometry and appearance by reconstructing a continuous radiance field for pure view synthesis. In contrast, Deep Reflectance Volumes explicitly express scene geometry and reflectance, and reproduce accurate high-frequency specularities and hard shadows. Ours is the first comprehensive neural rendering framework that enables both relighting and view synthesis with complex shading effects.

4.3 Rendering with Deep Reflectance Volumes

Unlike a mesh that is comprised of points with complex connectivity, a volume is a regular 3D grid, suitable for convolutional operations. Volumes have been widely used in deep learning frameworks for 3D applications [151, 157]. However, previous neural volumetric representations have only represented pixel colors; this can be used for view synthesis [86, 162], but does not support relighting or scene editing. Instead, we propose to jointly learn geometry and reflectance (i.e. material parameters) volumes to enable broader rendering applications including view synthesis, relighting and material editing in a comprehensive framework. Deep Reflectance Volumes are learned from a deep network and used to render images in a fully differentiable end-to-end process as shown in Fig. 4.2. This is made possible by a new differentiable volume ray marching module, which is motivated by physically-based volume rendering. In this section, we introduce our volume rendering method and volumetric scene representation. We discuss how we learn these volumes from unstructured images in Sec. 4.4.

4.3.1 Volume rendering overview

In general, volume rendering is governed by the physically-based volume rendering equation (radiative transfer equation) that describes the radiance that arrives at a camera [90, 101]:

$$L(\boldsymbol{c}, \boldsymbol{\omega}_o) = \int_0^{\infty} \tau(\boldsymbol{c}, \boldsymbol{x})[L_e(\boldsymbol{x}, \boldsymbol{\omega}_o) + L_s(\boldsymbol{x}, \boldsymbol{\omega}_o)]dx, \tag{4.1}$$

This equation integrates emitted, L_e, and in-scattered, L_s, light contributions along the ray starting at camera position $\boldsymbol{c}$ in the direction $-\boldsymbol{\omega}_o$. Here, x represents distance along the ray, and $\boldsymbol{x} = \boldsymbol{c} - x\boldsymbol{\omega}_o$ is the corresponding 3D point. $\tau(\boldsymbol{c}, \boldsymbol{x})$ is the transmittance factor that governs the loss of light along the line segment between $\boldsymbol{c}$ and $\boldsymbol{x}$:

$$\tau(\boldsymbol{c}, \boldsymbol{x}) = e^{-\int_0^x \sigma_t(z)dz}, \tag{4.2}$$

where $\sigma_t(z)$ is the extinction coefficient at location z on the segment. The in-scattered contribution is defined as:

$$L_s(\boldsymbol{x}, \boldsymbol{\omega}_o) = \int_{\mathscr{S}} f_p(\boldsymbol{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i) L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i, \tag{4.3}$$

in which $\mathscr{S}$ is a unit sphere, $f_p(\boldsymbol{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ is a generalized (unnormalized) phase function that expresses how light scatters at a point in the volume, and $L_i(\boldsymbol{x}, \boldsymbol{\omega}_i)$ is the incoming radiance that arrives at $\boldsymbol{x}$ from direction $\boldsymbol{\omega}_i$.

In theory, fully computing $L(\boldsymbol{c}, \boldsymbol{\omega}_o)$ requires multiple-scattering computation using Monte Carlo methods [101], which is computationally expensive and unsuitable for deep learning techniques. We consider a simplified case with a single point light, single scattering and no volumetric emission. The transmittance between the scattering location and the point light is handled the same way as between the scattering location and camera. The generalized phase function $f_p(\boldsymbol{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ becomes a reflectance function $f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i, \boldsymbol{n}(\boldsymbol{x}), R(\boldsymbol{x}))$ which computes reflected radiance at $\boldsymbol{x}$ using its local surface normal $\boldsymbol{n}(\boldsymbol{x})$ and the reflectance parameters $R(\boldsymbol{x})$ of a given surface reflectance model. Therefore, Eqn. 4.1 and Eqn. 4.3 can be simplified and

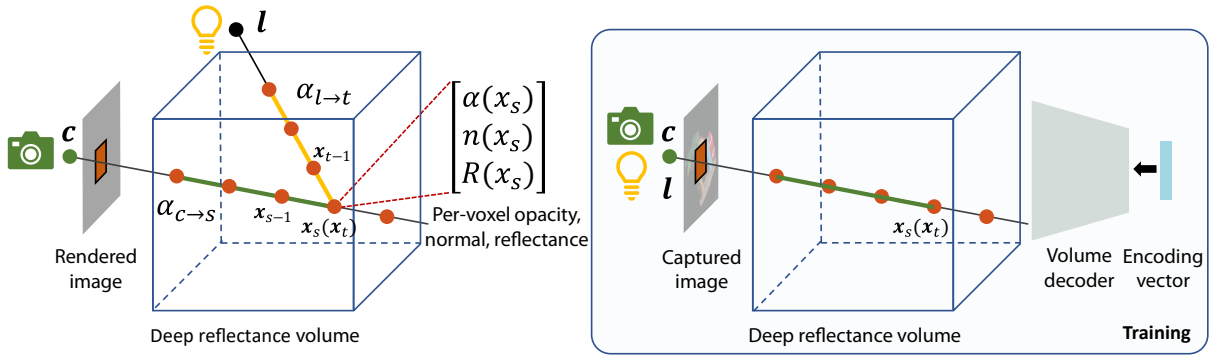

Figure 4.2. We propose Deep Reflectance Volume representation to capture scene geometry and appearance, where each voxel consists of opacity α, normal n and reflectance (material coefficients) R. During rendering, we perform ray marching through each pixel and accumulate contributions from each point $\boldsymbol{x}_s$ along the ray. Each contribution is calculated using the local normal, reflectance and lighting information. We accumulate opacity from both the camera $\alpha_{c\to s}$ and the light $\alpha_{l\to t}$ to model the light transport loss in both occlusions and shadows. To predict such a volume, we start from an encoding vector, and decode it into a volume using a 3D convolutional neural network; thus the combination of the encoding vector and network weights is the unknown variable being optimized (trained). We train on images captured with collocated camera and light by enforcing a loss function between rendered images and training images.

written concisely as [73, 90]:

$$L(\boldsymbol{c},\boldsymbol{\omega}_o) = \int_0^\infty \tau(\boldsymbol{c},\boldsymbol{x})\tau(\boldsymbol{x},\boldsymbol{l}) f_r(\boldsymbol{\omega}_o,\boldsymbol{\omega}_i,\boldsymbol{n}(\boldsymbol{x}),R(\boldsymbol{x}))L_{\boldsymbol{l}}(\boldsymbol{x},\boldsymbol{\omega}_i)dx, \tag{4.4}$$

where $\boldsymbol{l}$ is the light position, $\boldsymbol{\omega}_i$ corresponds to the direction from $\boldsymbol{x}$ to $\boldsymbol{l}$, $\tau(\boldsymbol{c},\boldsymbol{x})$ still represents the transmittance from the scattering point $\boldsymbol{x}$ to the camera $\boldsymbol{c}$, the term $\tau(\boldsymbol{x},\boldsymbol{l})$ (that was implicitly involved in Eqn. 4.3) is the transmittance from the light $\boldsymbol{l}$ to $\boldsymbol{x}$ and expresses light extinction before scattering, and $L_{\boldsymbol{l}}(\boldsymbol{x},\boldsymbol{\omega}_i)$ represents the light intensity arriving at $\boldsymbol{x}$ without considering light extinction.

4.3.2 A discretized, differentiable volume rendering module

To make volume rendering practical in a learning framework, we further approximate Eqn. 4.4 by turning it into a discretized version, which can be evaluated by ray marching [73, 90, 144]. This is classically expressed using opacity compositing, where opacity α is used to

represent the transmittance with fixed ray marching step size Δx. Points are sequentially sampled along a given ray, $\boldsymbol{\omega}_o$ from the camera position, $\boldsymbol{c}$ as:

$$\boldsymbol{x}_s = \boldsymbol{x}_{s-1} - \boldsymbol{\omega}_o \Delta x = \boldsymbol{c} - s\boldsymbol{\omega}_o \Delta x. \tag{4.5}$$

The radiance L_s and opacity $\alpha_{c\to s}$ along this path, $c \to s$, are recursively accumulated until $\boldsymbol{x}_s$ exits the volume as:

$$L_s = L_{s-1} + [1 - \alpha_{c\to(s-1)}][1 - \alpha_{l\to(t-1)}]\alpha(\boldsymbol{x}_s)L(\boldsymbol{x}_s), \tag{4.6}$$

$$\alpha_{c\to s} = \alpha_{c\to(s-1)} + [1 - \alpha_{c\to(s-1)}]\alpha(\boldsymbol{x}_s), \tag{4.7}$$

$$L(\boldsymbol{x}_s) = f_r(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i, \boldsymbol{n}(\boldsymbol{x}_s), R(\boldsymbol{x}_s))L_{\boldsymbol{l}}(\boldsymbol{x}_s, \boldsymbol{\omega}_i). \tag{4.8}$$

Here, $L(\boldsymbol{x}_s)$ computes the reflected radiance from the reflectance function and the incoming light, $\alpha_{c\to s}$ represents the accumulated opacity from the camera $\boldsymbol{c}$ to point $\boldsymbol{x}_s$, and corresponds to $\tau(\boldsymbol{c},\boldsymbol{x})$ in Eqn 4.4. $\alpha_{l\to t}$ represents the accumulated opacity from the light $\boldsymbol{l}$—i.e., $\tau(\boldsymbol{x},\boldsymbol{l})$ in Eqn. 4.4—and requires a *separate* accumulation process over samples along the $\boldsymbol{l} \to \boldsymbol{x}_s$ ray, similar to Eqn. 4.7:

$$\boldsymbol{x}_s = \boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \boldsymbol{\omega}_i \Delta x = \boldsymbol{l} - t\boldsymbol{\omega}_i \Delta x, \tag{4.9}$$

$$\alpha_{l\to t} = \alpha_{l\to(t-1)} + [1 - \alpha_{l\to(t-1)}]\alpha(\boldsymbol{x}_t). \tag{4.10}$$

In this rendering process (Eqn. 4.5-4.10), a scene is represented by an opacity volume α, a normal volume $\boldsymbol{n}$ and a BRDF volume R; together, these express the geometry and reflectance of the scene, and we refer to them as *Deep Reflectance Volumes*. The simplified opacity volume α is essentially one minus the transmission τ (depending on the physical extinction coefficient σ_t) over a ray segment of a fixed step size Δx; this means that α is dependent on Δx.

Our physically-based ray marching is fully differentiable, so it can be easily incorporated

in a deep learning framework and backpropagated through. With this rendering module, we present a neural rendering framework that simultaneously learns scene geometry and reflectance from captured images.

We support any differentiable reflectance model f_r and, in practice, use the simplified Disney BRDF model [67] that is parameterized by diffuse albedo and specular roughness (please refer to the supplementary materials for more details). Our opacity volume is a general geometry representation, accounting for both occlusions (view opacity accumulation in Eqn. 4.7) and shadows (light opacity accumulation in Eqn. 4.10). We illustrate our neural rendering with ray marching in Fig. 4.2. Note that, because our acquisition setup has collocated camera and lighting, $\alpha_{l\to t}$ becomes equivalent to $\alpha_{c\to s}$ during training, thus requiring only one-pass opacity accumulation from the camera. However, the learned opacity can still be used for re-rendering under any *non-collocated lighting* with two-pass opacity accumulation.

Note that while alpha compositing-based rendering functions have been used in previous work on view synthesis, their formulations are not physically-based [86] and are simplified versions that don't model lighting [125, 162]. In contrast, our framework is physically-based and models single-bounce light transport with complex reflectance, occlusions and shadows.

4.4 Learning Deep Reflectance Volumes

4.4.1 Overview

Given a set of images of a real scene captured under multiple known viewpoints with collocated lighting, we propose to use a neural network to reconstruct a Deep Reflectance Volume representation of a real scene. Similar to Lombardi et al. [86], our network starts from a 512-channel deep encoding vector that encodes scene appearance; in contrast to their work, where this volume only represents RGB colors, we decode a vector to an opacity volume α, normal volume $\boldsymbol{n}$ and reflectance volume R for rendering. Moreover, our scene encoding vector is not predicted by any network encoder; instead, we jointly optimize for a scene encoding vector

and scene-dependent decoder network.

Our network infers the geometry and reflectance volumes in a transformed 3D space with a learned warping function W. During training, our network learns the warping function W, and the geometry and reflectance volumes α_w, $\boldsymbol{n}_w$, R_w, where the subscript w refers to a volume in the warped space. The corresponding world-space scene representation is expressed by $V(\boldsymbol{x}) = V_w(W(\boldsymbol{x}))$, where V is α, $\boldsymbol{n}$ or R. In particular, we use bilinear interpolation to fetch a corresponding value at an arbitrary position $\boldsymbol{x}$ in the space from the discrete voxel values. We propose a decoder-like network, which learns to decode the warping function and the volumes from the deep scene encoding vector. We use a rendering loss between rendered and captured images as well as two regularizing terms.

4.4.2 Network architecture

Geometry and reflectance. To decode the geometry and reflectance volumes (α_w, $\boldsymbol{n}_w$, R_w), we use upsampling 3D convolutional operations to 3D-upsample the deep scene encoding vector to a multi-channel volume that contains the opacity, normal and reflectance. In particular, we use multiple transposed convolutional layers with stride 2 to upsample the volume, each of which is followed by a LeakyRelu activation layer. The network regresses an 8-channel $128 \times 128 \times 128$ volume that includes α_w, $\boldsymbol{n}_w$ and R_w—one channel for opacity α_w, three channels for normal $\boldsymbol{n}_w$, and four channels for reflectance R_w (three for albedo and one for roughness). These volumes express the scene geometry and reflectance in a transformed space, which can be warped to the world space for ray marching.

Warping function. To increase the effective resolution of the volume, we learn an affine-based warping function similar to [86]. The warping comprises a global warping and a spatially-varying warping. The global warping is represented by an affine transformation matrix W_g. The spatially varying warping is modeled in the inverse transformation space, which is represented by six basis affine matrices $\{W_j\}_{j=1}^{16}$ and a $32 \times 32 \times 32$ 16-channel volume B

that contains spatially-varying linear weights of the 16 basis matrices. Specifically, given a world-space position $\boldsymbol{x}$, the complete warping function W maps it into a transformed space by:

$$W(\boldsymbol{x}) = [\sum_{j=1}^{16} B_j(\boldsymbol{x}) W_j]^{-1} W_g \boldsymbol{x}, \tag{4.11}$$

where $B_j(\boldsymbol{x})$ represents the normalized weight of the jth warping basis at $\boldsymbol{x}$. Here, each global or local basis affine transformation matrix W_* is composed of rotation, translation and scale parameters, which are optimized during the training process. Our network decodes the weight volume B from the deep encoding vector using a multi-layer perceptron network with fully connected layers.

4.4.3 Loss function and training details

Loss function. Our network learns the scene volumes using a rendering loss computed using the differentiable ray marching process discussed in Sec. 4.3. During training, we randomly sample pixels from the captured images and do the ray marching (using known camera calibration) to get the rendered pixel colors L_k of pixel k; we supervise them with the ground truth colors $\tilde{L}_k$ in the captured images using a L_2 loss. In addition, we also apply regularization terms from additional priors similar to [86]. We only consider opaque objects in this work and enforce the accumulated opacity along any camera ray $\alpha_{c_k \to s'}$ (see Eqn. 4.7, here k denotes a pixel and s' reflects the final step that exits the volume) to be either 0 or 1, corresponding to a background or foreground pixel, respectively. We also regularize the per-voxel opacity to be sparse over the space by minimizing the spatial gradients of the logarithmic opacity. Our total loss function is given by:

$$\sum_k \|L_k - \tilde{L}_k\|^2 + \beta_1 \sum_k [\log(\alpha_{c_k \to s'}) + \log(1 - \alpha_{c_k \to s'})] + \beta_2 \sum \|\nabla_{\boldsymbol{x}} \log \alpha(\boldsymbol{x})\| \tag{4.12}$$

Here, the first part reflects the data term, the second regularizes the accumulated α and the third regularizes the spatial sparsity.

Training details. We build our volume as a cube located at $[-1,1]^3$. During training, we randomly sample 128×128 pixels from 8 captured images for each training batch, and perform ray marching through the volume using a step size of $1/64$. Initially, we set $\beta_1 = \beta_2 = 0.01$; we increase these weights to $\beta_1 = 1.0$, $\beta_2 = 0.1$ after 300000 iterations, which helps remove the artifacts in the background and recover sharp boundaries.

4.5 Results

In this section we show our results on real captured scenes. We first introduce our acquisition setup and data pre-processing. Then we compare against the state-of-the-art mesh-based appearance acquisition method, followed by a detailed analysis of the experiments. We also demonstrate material editing results with our approach. Please refer to the supplementary materials of the original publication for video results.

Data acquisition. Our approach learns the volume representation in a scene dependent way from images with collocated view and light; this requires adequately dense input images well distributed around a target scene to learn complete appearance. Such data can be practically acquired by shooting a video using a handheld cellphone; we show one result using this practical handheld setup in Fig. 4.4. For other results, we use a robotic arm to automatically capture more uniformly distributed images around scenes for convenience and thorough evaluations; this allows us to evaluate the performance of our method with different numbers of input images that are roughly uniformly distributed as shown in Tab. 4.1. In the robotic arm setups, we mount a Samsung Galaxy Note 8 cellphone to the robotic arm and capture about 480 images using its camera and the built-in flashlight in a dark room; we leave out a subset of 100 images for validation purposes and use the others for training. We use the same phone to capture a 4-minute video of the object in CAPTAIN and select one image for training for every 20 frames, which

effectively gives us 310 training images.

Data pre-processing. Our captured objects are roughly located around the center of the images. We select one fixed rectangular region around the center that covers the object across all frames and use it to crop the images as input for training. The resolution of the cropped training images fed to our network ranges from 400×500 to 1100×1100. Note that we do not use a foreground mask for the object. Our method leverages the regularization terms in training (see Sec. 4.4.3), which automatically recovers a clean background. We calibrate the captured images using structure from motion (SfM) in COLMAP [114] to get the camera intrinsic and extrinsic parameters. Since SfM may fail to register certain views, the actual number of training images varies from 300 to 385 in different scenes. We estimate the center and bounding box of the captured object with the sparse reconstructions from SfM. We translate the center of the object to the origin and scale it to fit into the $[-1,1]^3$ cube.

Implementation and timing. We implement our system (both neural network and differentiable volume rendering components) using PyTorch. We train our network using four NVIDIA 2080Ti RTX GPUs for about two days (about 450000 iterations; though 200000 iterations for 1 day typically already converges to good results, see Fig. 4.5). At inference time, we directly render the scene from the reconstructed volumes without the network. It takes about 0.8s to render a 700×700 image under collocated view and light. For non-collocated view and light, the rendering requires connecting each shading point to the light source with additional light-dependent opacity accumulation, which is very expensive if done naively. To facilitate this process, we perform ray marching from the light's point of view and precompute the accumulated opacity at each spatial position of the volume. During rendering, the accumulated opacity for the light ray can be directly sampled from the precomputed volume. By doing so, our final rendering under arbitrary light and view takes about 2.3s.

Comparisons with mesh-based reconstruction. We use a practical acquisition setup where we capture unstructured images using a mobile phone with its built-in flashlight on

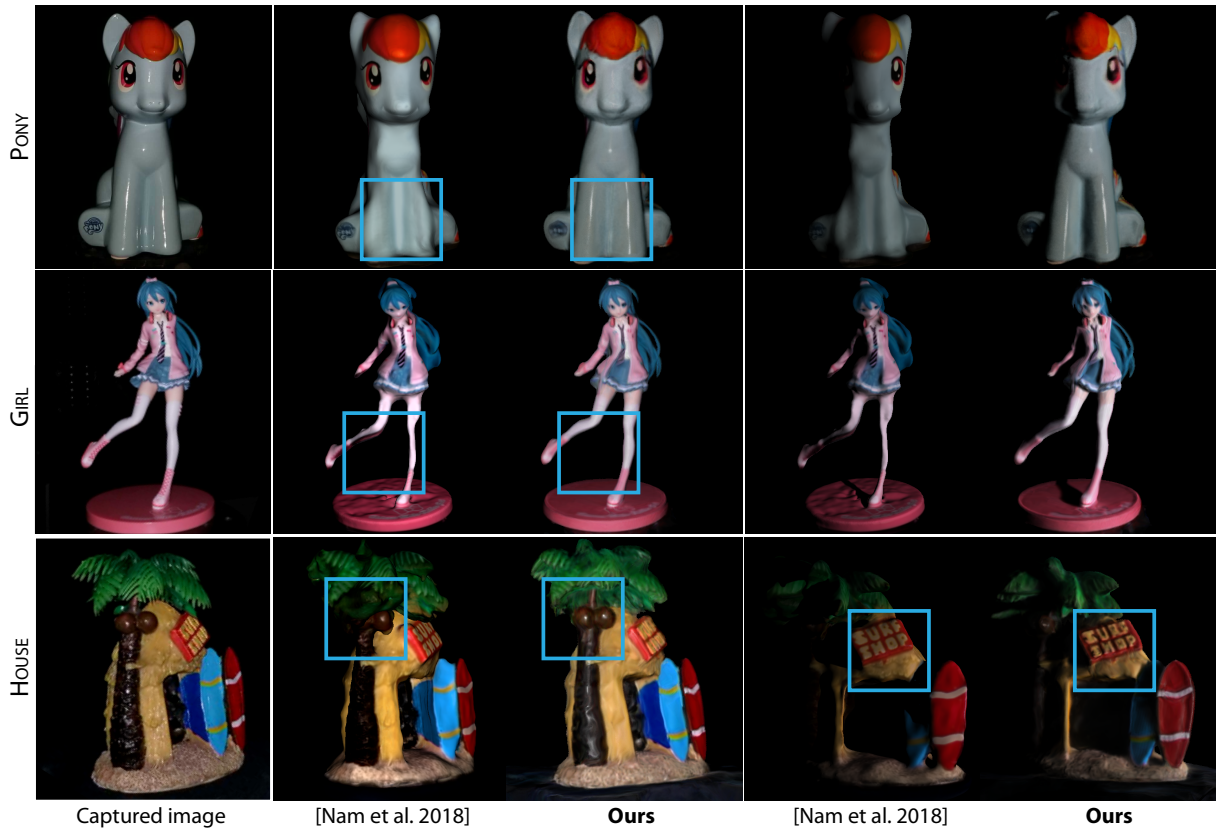

Figure 4.3. Comparisons with mesh-based reconstruction. We show renderings of the captured object under both collocated (column 2, 3) and non-collocated (column 4, 5) camera and light. We compare our volume-based neural reconstruction against a state-of-the-art method [97] that reconstructs mesh and per-vertex BRDFs. Nam et al. [97] fails to handle such challenging cases and recovers inaccurate geometry and appearance. In contrast our method produces photo-realistic results.

in a dark room. Such a mildly controlled acquisition setup is rarely supported by previous works [16, 65, 152, 153, 155, 163]. Therefore, we compare with the state-of-the-art method proposed by Nam et al. [97] for mesh-based geometry and reflectance reconstruction, that uses the same cellphone setup as ours to reconstruct a mesh with per-vertex BRDFs, and supports both relighting and view synthesis. Figure 4.3 shows comparisons on renderings under both collocated and non-collocated view-light conditions. The comparison results are generated from the same set of input images, and we requested the authors of [97] run their code on our data and compared on the rendered images provided by the authors. Please refer to the supplementary materials of the original publication for video comparisons.

Table 4.1. We evaluate the performance of our method on the HOUSE scene with different numbers of training images. Although we use all 385 images in our final experiments, our method is able to achieve comparable performance with as few as 200 images for this challenging scene.

	25	50	100	200	385
PSNR	25.33	26.36	26.95	27.85	28.13
SSIM	0.70	0.73	0.75	0.80	0.81

Table 4.2. We compare against DeepVoxels on synthesizing novel views under collocated lights and report the PSNR/SSIM scores. The results show that our method generates more accurate renderings. Note that we retrain our model with a resolution of 512×512 for a fair comparison.

	HOUSE	CARTOON
[124]	0.786/25.81	0.532/16.34
Ours	**0.896/30.44**	**0.911/29.14**

As shown in Fig. 4.3, our results are significantly better than the mesh-based method in terms of both geometry and reflectance. Note that, Nam et al. [97] leverage a state-of-the-art MVS method [115] to reconstruct the initial mesh from captured images and performs an optimization to further refine the geometry; this however still fails to recover the accurate geometry in texture-less, specular and thin-structured regions in those challenging scenes, which leads to seriously distorted shapes in PONY, over-smoothness and undesired structures in HOUSE, and degraded geometry in GIRL. Our learning-based volumetric representation avoids these mesh-based issues and models the scene geometry accurately with many details. Moreover, it is also very difficult for the classical per-vertex BRDF optimization in [97] to recover high-frequency specularities, which leads to over-diffuse appearance in most of the scenes; this is caused by the lack of constraints for the high-frequency specular effects, which appear in very few pixels in limited input views. In contrast, our optimization is driven by our novel neural rendering framework with deep network priors, which effectively correlates the sparse specularities in different regions through network connections and recovers realistic specularities and other appearance effects.

Comparison on synthesizing novel views. We also make a comparison on synthesizing novel views under collocated lights against a view synthesis method DeepVoxels [124], which

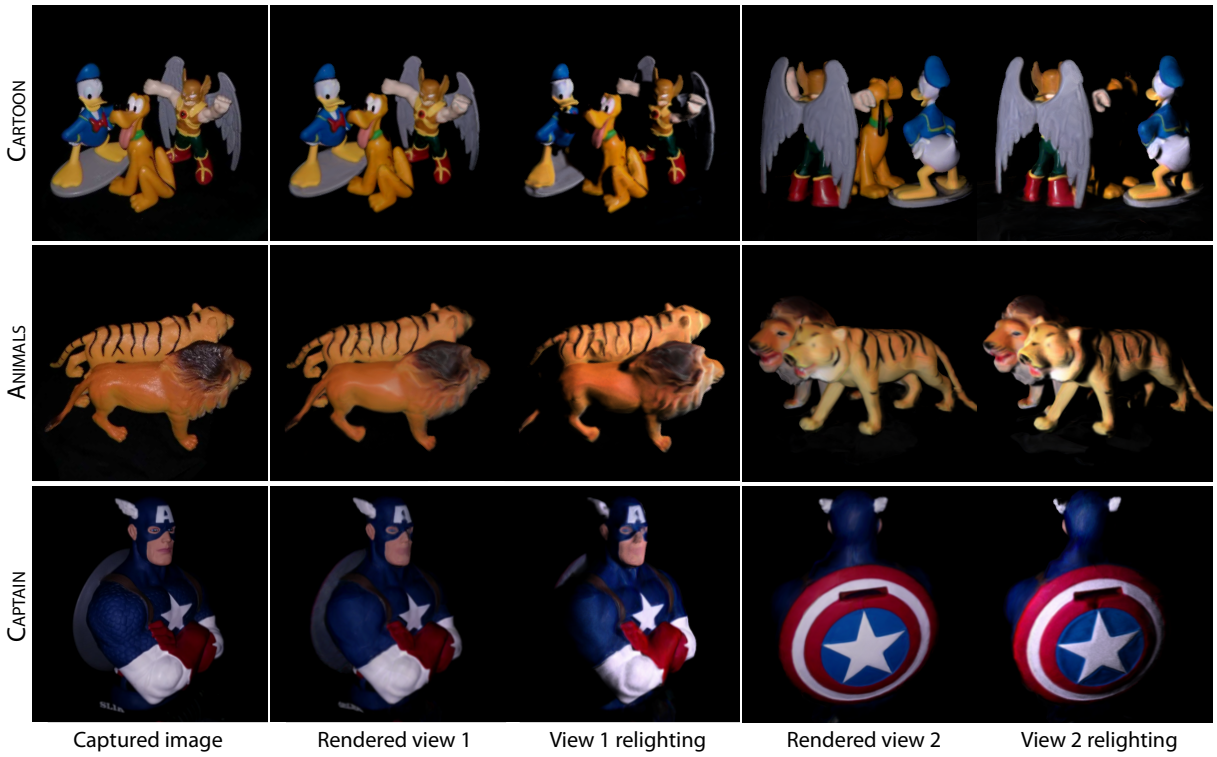

Figure 4.4. Additional results on real scenes. We show renderings under novel view and lighting conditions. Our method is able to handle scenes with multiple objects (top two rows) and model the complex occlusions between them. Our method can also generate high-quality results from casual handheld video captures (third row), which demonstrates the practicability of our approach.

encodes view-dependent appearance in a learnt 3D-aware neural representation. Note that DeepVoxels does not support relighting. As shown in Fig. 4.2, our method is able to generate renderings of higher quality with higher PSNR/SSIM scores. In contrast, DeepVoxels fails to reason about the complex geometry in our real scenes, thus resulting in degraded image quality. Please refer to the supplementary materials for visual comparison results.

Additional results. We show additional relighting and view synthesis results of complex real scenes in Fig. 4.4. Our method is able to handle scenes with multiple objects, as shown in scene CARTOON and ANIMALS. Our volumetric representation can accurately model complex occlusions between objects and reproduce realistic cast shadows under novel lighting, which are never observed by our network during the training process. In the CAPTAIN scene, we show

the result generated from *handheld* mobile phone captures. We select frames from the video at fixed intervals as training data. Despite the potential existence of motion blur and non-uniform coverage, our method is able to generate high-quality results, which demonstrates the robustness and practicality of our approach. Please refer to the supplementary materials of the original publication for video results.

Evaluation of the number of inputs. Our method relies on an optimization over adequate input images that capture the scene appearance across different view/light directions. We evaluate how our reconstruction degrades with the decrease of training images on the HOUSE scene. We uniformly select a subset of views from the full training images and train our model on them. We evaluate the trained model on the test images, and report the SSIMs and PSNRs in Fig. 4.1. As we can see from the results, there is an obvious performance drop when there are fewer than 100 training images due to insufficient constraints. On the other hand, while we use the full 385 images for our final results, our method in fact achieves comparable performance with only 200 for this scene, as reflected by their close PSNRs and SSIMs.

Comparison with direct optimization. Our neural rendering leverages a "deep volume prior" to drive the volumetric optimization process. To justify the effectiveness of this design, we compare with a naive method that directly optimizes the parameters in each voxel and the warping parameters using the same loss function. We show the optimization progress in Fig. 4.5. Note that, the naive method converges significantly slower than ours, where the independent voxel-wise optimization without considering across-voxel correlations cannot properly disentangle the ambiguous information in the captured images; yet, our deep optimization is able to correlate appearance information across the voxels with deep convolutions, which effectively minimizes the reconstruction loss.

Material editing. Our method learns explicit volumes with physical meaning to represent the reflectance of real scenes. This enables broad image synthesis applications like editing the materials of captured scenes. We show one example in Fig. 4.6, where we successfully make

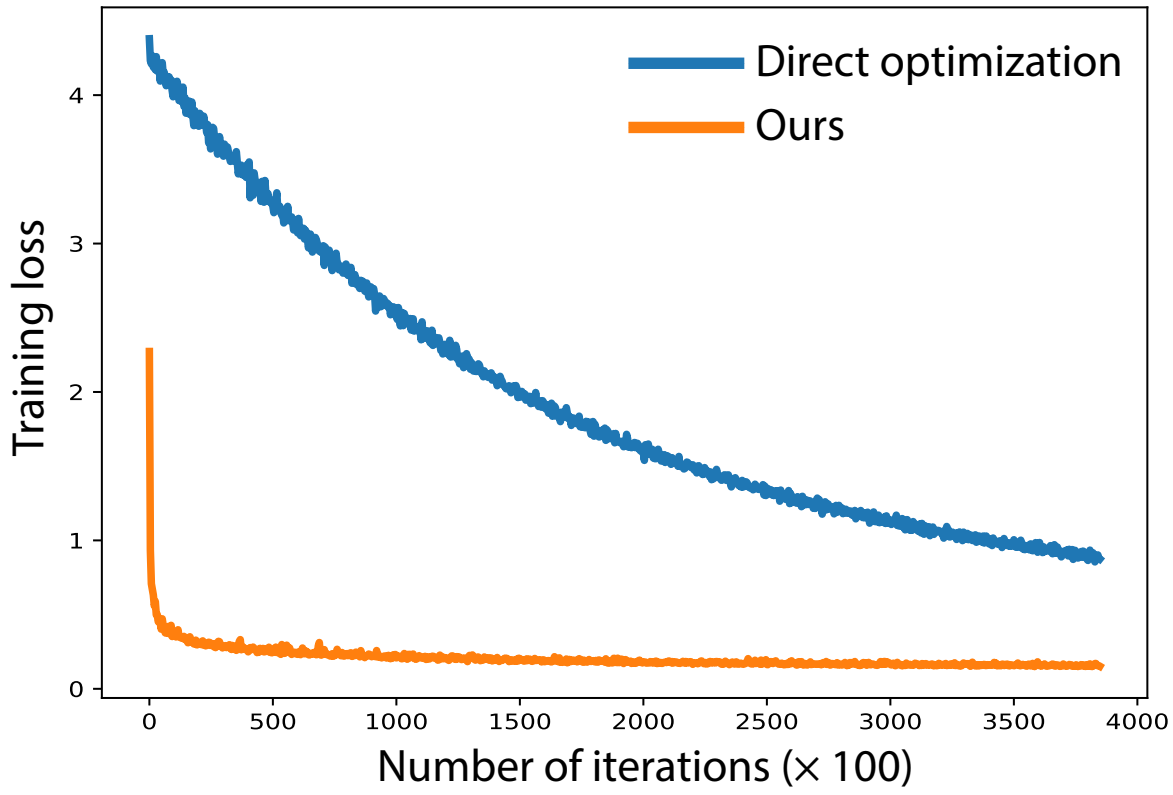

Figure 4.5. We compare our deep prior based optimization against direct optimization of the volume and warping function without using networks. Direct optimization converges significantly slower than our method, which demonstrates the effectiveness of regularization by the networks.

the scene glossier by decreasing the learned roughness in the volume. Note that, the geometry and colors are still preserved in the scene, while novel specularities are introduced which are not part of the material appearance in the scene. This example illustrates that our network disentangles the geometry and reflectance of the scene in a reasonable way, thereby enabling sub-scene component editing without influencing other components.

Limitations. We reconstruct the deep reflectance volumes with a resolution of 128^3, which is restricted by available GPU memory. While we have applied a warping function to increase the actual utilization of the volume space, and demonstrated that it is able to generate compelling results on complex real scenes, it may fail to fully reproduce the geometry and appearance of scenes with highly complex surface normal variations and texture details. Increasing the volume resolution may resolve this issue. In the future, it would also be interesting to investigate

Figure 4.6. Our approach supports intuitive editing of the material properties of a captured object. In this example we decrease the roughness of the object to make it look like glossy marble instead of plastic.

how to efficiently apply sparse representations such as octrees in our framework to increase the capacity of our volume representation. The current reflectance model we are using is most appropriate for opaque surfaces. Extensions to other materials like hair, fur or glass could be potentially addressed by applying other reflectance models in our neural rendering framework.

4.6 Conclusion

We have presented a novel approach to learn a volume representation that models both geometry and reflectance of complex real scenes. We predict per-voxel opacity, normal, and reflectance from unstructured multi-view mobile phone captures with the flashlight. We also introduce a physically-based differentiable rendering module to enable renderings of the volume under arbitrary viewing and lighting directions. Our method is practical, and supports novel view synthesis, relighting and material editing, which has significant potential benefits in scenarios such as 3D visualization and VR/AR applications.

This chapter is based on the material as it appears in European Conference on Computer Vision (ECCV), 2020 ("Deep Reflectance Volumes: Relightable Reconstructions from Multi-View Photometric Images", Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, Ravi Ramamoorthi). The dissertation author was the primary investigator and author of this paper.

Chapter 5

Deep Relightable Appearance Models for Animatable Faces

5.1 Introduction

In the prior chapters, we have presented methods to faithfully reproduce the appearance of static objects. In this chapter, we take one step further and propose novel methods to model the dynamic appearance of human faces and create animatable and relightable avatars. Avatar creation has seen a notable increase in the use of learning-based techniques in recent years [85, 116, 96] Traditional physically-inspired methods [119, 141] require precise geometry and reflectance, where costly and time-consuming manual cleanup is typically needed. In contrast, learning-based methods use general function approximators in the form of deep neural networks to faithfully model the appearance of human faces. They can achieve impressive realism with completely automated pipelines without relying on precise estimates of face geometry and material properties. They can also exhibit an efficient functional form that enables real-time generation and rendering in demanding applications such as VR [85], where classical ray-tracing methods can be too computationally intensive [141].

Despite their many advantages, avatars created using learning based techniques have so far been limited to a single lighting condition [85, 96]. For example, Lombardi et al. build avatars that support novel viewpoints and expressions, but their model is limited to the uniform lighting condition under which the data was captured. Although there has been great progress

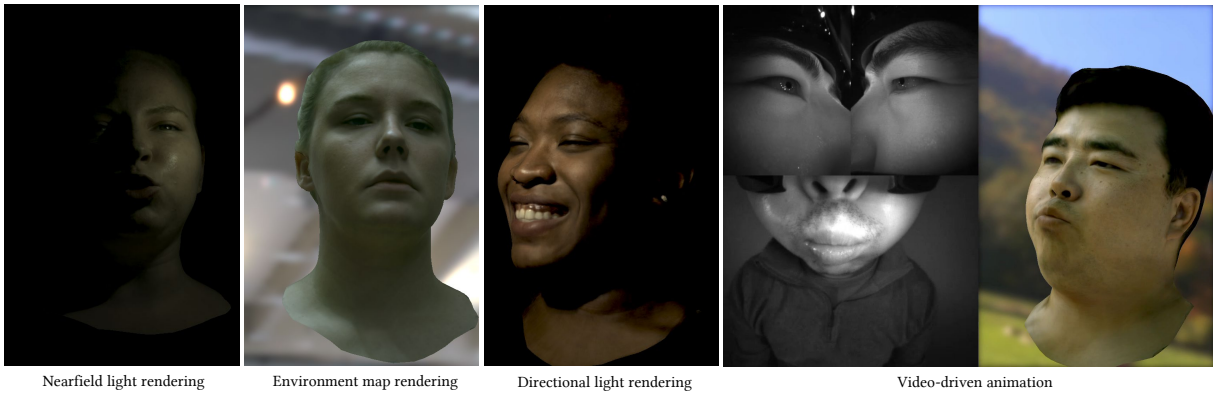

Figure 5.1. Our relightable facial appearance model supports renderings under novel viewpoints, expressions, and lighting conditions including nearfield lighting, directional lighting, and environment lighting. Our model is also animatable and can be driven by images captured from cameras on head-mounted displays.

in learning-based relighting, existing methods are limited to 2D images [129, 155], static scenes [129, 155, 159], or performance replay [91], which are not suitable for generating dynamic renderings under novel expressions and lighting conditions (see Table 5.1). This limitation has prevented the broader adoption of learning-based avatars in game and film production, where consistency between character and environment is essential.

In this work, we describe Deep Relightable Appearance Models (DRAM), a learning-based method for building relightable avatars. Our model supports rendering under novel viewpoints, novel expressions and more importantly, it can be rendered under novel lighting conditions, where we can reconstruct complex visual phenomena such as specularities, glints and subsurface scattering. We build the relightable model from light-stage captures of dynamic performances under a sparse set of space- and time-multiplexed illumination patterns. Like [85], we train our model using the variational auto-encoder framework [71], which produces a well structured latent space of expressions, suitable for animation. To avoid overfitting to the lighting conditions observed during capture, we leverage the additive property of light transport [19] and generate expression- and view-dependent textures for each light in the scene, which are then fused with intensity-defined weights into the final lit texture. Since the lighting information is fed at a later stage of the decoder network, instead of at its bottleneck, we call this model a

late-conditioned model. It affords generalization to completely unseen lighting environments including both distant directional lighting and real environment maps (Figure 5.1), and it exhibits smooth interpolation of point light sources despite the discrete set of 460 lights used during capture. Finally, it can generate compelling near-field illumination effects (Figure 5.8), which is particularly challenging for a learning-based approach that exclusively uses data with distant light sources.

Although late-conditioned DRAM (DRAM_ℓ) exhibits good generalization properties, its architecture is not suitable for real-time applications, since each point light in the scene requires the generation of a light-specific texture. For natural environments, the large number of illuminating directions make it computationally prohibitive to generate. This limitation is shared by many previous works [91, 159, 30]. However, we observe that early-conditioned deep neural networks that input the desired lighting condition at the network's bottleneck can exhibit enough capacity to model the span of a single person's illuminated facial appearances while being considerably more efficient to evaluate.

The main drawback of *early-conditioned* models is their poor extrapolation properties to unseen natural illumination conditions. Thus, we use DRAM_ℓ to generate renderings of the face under a large number of natural illumination conditions, which we then use to train an efficient early-conditioned model, obviating the need for it to extrapolate to those conditions during test time. We call this model early-conditioned DRAM (DRAM_ε) and propose a hyper-network architecture for its representation. It comprises two components, one network that takes the desired lighting condition as input and predicts the weights for a second network that produces the view, expression and lighting-dependent texture. Such a design further increases the capacity of the network and results in renderings of much higher quality while maintaining a low computational cost. The result is a method for creating animatable faces that can be relit using novel illumination conditions and rendered in real time. We demonstrate a use case of our relightable model by live-driving it from a VR-headset mounted camera [138] and rendering in novel and varying illumination (Figure 5.1).

Table 5.1. Feature comparison with previous methods. Ours is the only approach that enables a relightable and animatable model, in addition to free viewpoint and dynamic expressions.

	Free viewpoint	Relightable	Dynamic capture	Animatable
Wenger et al.[139]	✗	✓	✓	✗
Lombardi et al. [85]	✓	✗	✓	✓
Xu et al. [155]	✗	✓	✗	✗
Meka et al. [91]	✗	✓	✓	✗
Sun et al. [129]	✗	✓	✗	✗
Sun et al. [130]	✗	✓	✗	✗
Zhang et al. [159]	✓	✓	✗	✗
Meka et al. [92]	✓	✓	✓	✗
Ours	✓	✓	✓	✓

A summary of the contributions of this work are:

- A method for generating high-fidelity animatable personalized face avatars from dynamic multi-view light-stage data that can be relit under novel lighting environments, including challenging natural illumination and near-field lighting that are far from what is observed during training.
- A student-teacher framwork for training an efficient relighting model that achieves real-time rendering while overcoming generalization limitations typically exhibited by such models.
- A novel hyper-network architecture for early-conditioned models that achieves significantly improved reconstruction accuracy while remaining efficient to evaluate.
- The first demonstration of relightable faces driven by headset mounted cameras for VR applications.

5.2 Related Works

Face modeling. Traditional methods for face modeling [6, 119] depend on precise 3D reconstruction of human faces, which requires a large amount of manual effort and is not suitable

for real-time applications. Recently Lombardi et al. [85] propose a data-driven method for face modeling. It applies a conditional variational autoencoder to learn a latent representation for facial expressions and regresses a tracked mesh and a view-dependent texture to model the appearance of human faces. Schwartz et al. [116] build on the same framework and explicitly model human eyes for better eye contact. However, these methods are limited to a single lighting condition and do not support relighting under novel lighting conditions.

Reflectance acquisition. To relight human faces under novel lighting conditions, previous approaches have tried to estimate the reflectance properties of human faces from captured images. Such methods usually assume a simplified reflectance model based on physical priors. Some previous works develop their method based on the diffuse assumption for faces. Garrido et al. [43] and Cal et al. [20] assume that faces are diffuse and jointly estimate the diffuse albedo and facial geometry from monocular videos. Shu et al. [121] applies a learning-based method to infer facial normals and albedo from a single image. Other works also model specular reflections of human faces. Both diffuse and specular albedos are estimated from captures with different acquisition setups such as spherical gradient illuminations [88, 49] and multi-view captures under passive illumination [46]. Yamaguchi et al. [156] applies a deep-learning based approach to infer both reflectance and high-frequency displacement maps to model mesoscopic surface details on human faces from a single RGB image under uncontrolled illuminations. More complex reflectance models that consider subsurface scatterings have also been applied. Jensen et al. [59] introduced a bidirectional surface scattering model for human faces based on a dipole diffusion approximation and proposed a method to measure the model parameters. Ghosh et al. [44] recover layered facial reflectance including specular reflectance and scatterings at different layers from a set of twenty photographs under environmental and projected illuminations. All these physically-based approaches can only model a portion of face appearances, and fail to faithfully reproduce the complex visual appearance of human faces, especially for dynamic animations, where different expressions will result in significant differences in appearance. In

addition, rendering with such reflectance models under complex lighting conditions also requires physically-based path tracers, which is computationally expensive and not suitable for real-time applications.

Image-based relighting. Methods in this category make use of the linearity of light transport and synthesize renderings of the scene under novel lighting conditions by combining images under a set of basis lighting patterns. A category of these works focus on the relighting of static scenes. Debevec et al. [30] captures the reflectance field of human faces by capturing images under a dense sampling of directional incident illuminations. Xu et al. [155] propose a learning-based method to synthesize renderings of static scenes at a novel lighting direction from a sparse set of captures. Sun et al. [129] train a network to directly regress the relighting results under novel environment lightings from a single portrait image. Their results have limited fidelity and cannot recover visual effects such as specularities and detailed glints. In a later work [130], they propose a method to increase the resolution of static light stage captures and enable relighting under an arbitrary lighting direction. Zhang et al. [159] achieves free-viewpoint relighting of static human captures by explicitly reconstructing the geometry of the scenes and training a network to synthesize texture-space RGB images under desired view and lighting direction. Most of these methods require a static capture setup, where the subject remains still and maintains a fixed expression while a one-light-at-a-time (OLAT) capture is performed. This limits their ability to capture transient expressions of the face in motion required for building dynamic animatable face avatars.

Free-viewpoint Methods Some exising works leverage 3D reconstructions of the scene to enable free-viewpoint rendering of the relightable models they build. Wenger et al. [139] achieves dynamic relighting with time-multiplexed lightings where the subject is illuminated with a rapid series of basis lighting patterns. They warp adjacent frames to the target frame using optical flow so as to relight the target frame, which suffers from potential misalignments due to inaccuracies in flow computation. Meka et al. [91] applies colored gradient illumination for

efficient dynamic captures, and they train a network to infer the renderings under an arbitrary lighting direction from two gradient illumination captures, which are then used for relighting under novel lighting conditions. Their method requires colored illuminations, and suffers from misalignments between the two input captures. In a follow-up work [92], they reconstruct the geometry of the human and propose a network to regress dynamic textures under an arbitrary lighting direction from color gradient captures, which enables free-viewpoint dynamic relighting. However, they use per-frame reconstructions without correspondence between the frames, which makes it unsuitable for novel sequence animation, restricting its use to performance replay.

We provide a detailed comparison of features between previous methods and our method in Table 5.1. Compared to previous works, our method is the first that supports novel viewpoints, novel lighting conditions, dynamic playback and animation.

5.3 Data Acquisition

The appearance of human faces can be modeled as a function of the facial expression, viewpoint and lighting condition. We propose to use neural networks to approximate such a function. To supervise the training of such a network, ideally we could capture image data of all possible combinations of these three factors using a light stage. Our capture system consists of $\sim$140 color cameras and 460 white LED lights. All the LEDs can be independently controlled with adjustable lighting intensity. The cameras and lights are positioned on a spherical dome with a radius of 1.1m surrounding the captured subject.

To densely sample expression and viewpoint combinations, a capture-subject is asked to make a predefined set of facial expressions, recite a set of 50 phonetically balanced sentences, perform a range-of-motion sequence, and have a short natural conversation with a colleague [85]. During captures, all the 140 cameras synchronously capture at a frame rate of 90 frames per second, and output 8-bit Bayer-pattern color images with a resolution of 2668×4096.

The simultaneous capture of images with different lighting conditions is much more chal-

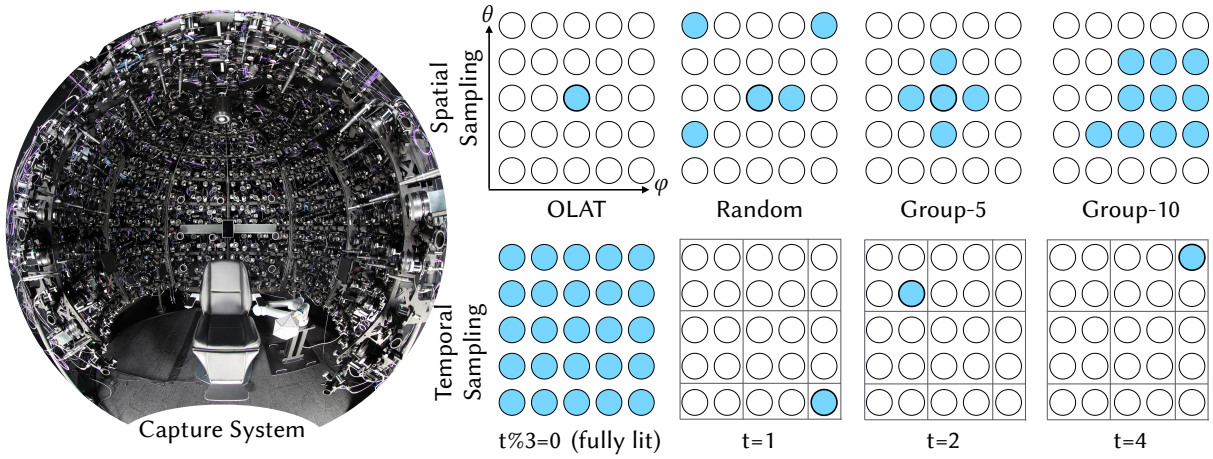

Figure 5.2. Capture system with lights and cameras in a spherical dome (left) and light patterns used during capture (right). We evaluate different spatial groupings: one-light-at-a-time, 5 random lights, and spatial groups of 5 and 10 lights. Temporally, we sample lights using stratified random sampling.

lenging in comparison. Wavelength multiplexed approaches [52, 47] are limited in the frequency bands that can be used, while time-multiplexed approaches [139, 143] present challenges in capturing dynamic content with transient expressions. Our work follows the approach of Wenger et al. [139], where time-multiplexed lighting is captured by rapidly cycling over a set of basis lighting patterns. However, instead of requiring static expressions for each cycle, we rely on amortized inference [72] to disentangle lighting from expression in our captures of the face in motion, and evaluate the suitability of different kinds of lighting patterns for this approach. Specifically, we evaluate the efficacy of OLAT, *Random* (i.e., spatially unstructured sets of 5 lights), and two sets of *Group* patterns (i.e., spatially clustered groups of lights); one with five lights and another with ten. The rank of the basis formed by each lighting pattern ranges from 460 to 50. In all cases, a fully-lit frame is interleaved every third frame to enable face tracking [146] which produces a topologically consistent mesh, $\mathbf{M} \in \mathbb{R}^{3\times 7306}$, for every frame[1]. In discussions that follow, we will use the following notation to refer to the lighting at a given

[1] In this work we presume the mesh between every third frame can be well approximated by linearly interpolating its adjacent tracked meshes.

frame:

$$L = \{b_1, b_2, ..., b_n\} \tag{5.1}$$

where b_i is the index of the i-th light that is turned on and n is the total number of lights for that frame.

The choice of lighting patterns we consider in this work is guided by a few factors that are difficult to meet simultaneously. First, it is desirable to see many different facial expressions for each lighting condition. OLAT generates the most complete set of lighting conditions with the finest spatial resolution, but has a long cycle time, minimizing the variety of facial expression seen in each lighting condition. Second, it is desirable to see many complementary lighting conditions for each facial expression. To achieve this, we temporally sample light directions using spatially stratified random sampling: lights are first stratified into 8 groups (represented as grid cells in Fig. 5.2) with the next group chosen using furthest-group sampling across consecutive frames, and the light direction chosen randomly within a group. Third, it is preferable to have as much light as possible to overcome the *noise floor* of our cameras. Random and grouped lights trade off the spatial granularity of each lighting condition, but increase the light available to the cameras, potentially relaxing requirements on the capture system.

5.4 Building Relightable Avatars

Our goal is to build personalized expressive face avatars that can be rendered from novel viewpoints and relit to match the lighting in novel environments. We leverage the representation power of neural networks to map viewpoint, expression and lighting to highly accurate texture and geometry, which can be used to synthesize an image using standard rasterization techniques [85]. To overcome challenges presented by dynamic capture that are discussed in §5.3, we leverage the amortized inference properties of conditional variational auto-encoders (CVAE) [71] to disentangle expression from lighting in our representation. However, a naive implementation of

such an architecture generalizes poorly to novel lighting conditions that one might encounter in practice. This includes natural indoor and outdoor illumination conditions that can be quite different from the point light patterns used during data capture. An example of such a failure is illustrated in Figure 5.14. A key contribution of our work is a two-stage system that enables efficient relightable models that generalize to unseen lighting conditions to be learned.

The first stage of our system comprises a representation, DRAM_ℓ, that achieves generalization by leveraging the additive property of light. Although it is computationally expensive to evaluate, it allows us to synthesize high fidelity face images under lighting conditions that are far from what can be captured in our light stage. Thus, we use $\text{DRAM}_\ell$to generate a large number of high-quality synthetic images to complement our real captured images, to overcome the need for the efficient neural network architectures used in the second stage to extrapolate to those conditions.

Armed with an expanded dataset generated from the first stage, the second stage of our system involves training a novel neural network architecture, DRAM_ε, with high capacity but low compute. Here, we employ a hyper-network that produces lighting-specific network weights of a standard deconvolutional architecture that has previously been demonstrated to be capable of spanning the space of expressions for a single lighting condition [85]. The resulting model attains real-time performance of 75 frames per second on a Nvidia Tesla V100, and we demonstrate its suitability for animation by driving it from headset-mounted cameras as discussed in Section 5.5.

For all the models we describe in this section, we follow the data preprocessing described in [85]. Specifically, images, $\mathbf{I} \in \mathbb{R}^{3\times 2668\times 4096}$, of a specific frame and camera viewpoint, whether real or synthetic, are unwarped into a texture, $\mathbf{T}\in R^{3\times 1024^2}$, using the tracked mesh, $\mathbf{M}$, for that frame. We also calculate the average texture, $\bar{\mathbf{T}}$, for every frame by averaging the texture at each camera, which is used as input to CVAE to encourage better disentanglement between viewpoint and latent space. Representative visualizations of these elements are shown in Figure 5.3.

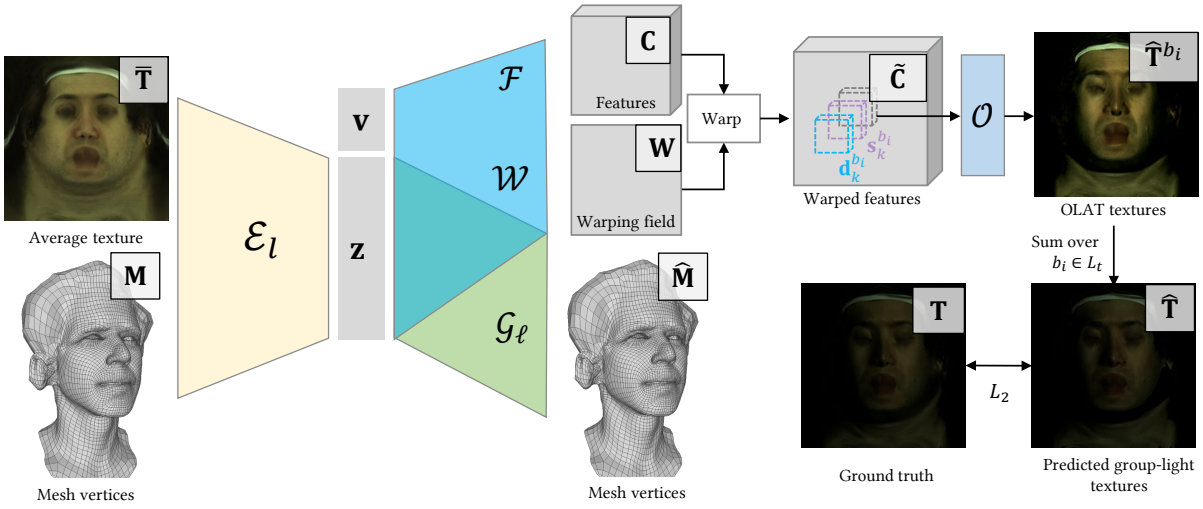

Figure 5.3. Network architecture for our late-conditioned model. Expression and view-dependent features are generated with an encoder-decoder architecture, and late-conditioned with in an MLP network to produce single-light textures, which can be modulated by light intensity and summed to produce more complex illuminations.

5.4.1 DRAM$_\ell$: A Late-conditioned Model

As shown in Figure 5.3, our late-conditioned model is a CVAE comprised of an encoder $\mathscr{E}_\ell$ and a decoder $\mathscr{D}_\ell$. The encoder takes the tracked mesh, $\mathbf{M}$, and the average texture, $\bar{\mathbf{T}}$, of its nearest fully-lit frame as input and outputs the parameters of its variational distribution, $\mathscr{N}$, from which the latent code $\mathbf{z} \in \mathbb{R}^{256}$ is sampled:

$$\boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow \mathscr{E}_\ell\left(\mathbf{M}, \bar{\mathbf{T}}\right) \quad , \quad \mathbf{z} \sim \mathscr{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2). \tag{5.2}$$

A Gaussian distribution with diagonal covariance is used for $\mathscr{N}$. The reparametrization trick [71] is used to ensure differentiability of the sampling process.

The input to the decoder $\mathscr{D}_\ell$ includes the latent vector $\mathbf{z}$, the view direction $\mathbf{v}$ of the camera relative to the head orientation in that frame, and the lighting condition L, transformed to the head coordinate system. The decoder outputs the reconstructed mesh $\hat{\mathbf{M}}$ and predicts the textures corresponding to each single light in L, which sum up to produce the final texture $\hat{\mathbf{T}}$. The decoder consists of two branches: the geometry branch, $\mathscr{G}_\ell$, which takes the latent vector

as input and predicts the mesh, and the texture branch, $\mathscr{T}_\ell$, which additionally conditions on viewpoint and lighting to produce texture:

$$\hat{\mathbf{M}} = \mathscr{G}_\ell(\mathbf{z}) \quad , \quad \hat{\mathbf{T}} = \mathscr{T}_\ell(\mathbf{z}, \mathbf{v}, L). \tag{5.3}$$

Our texture branch consists of three components; a feature network $\mathscr{F}$, a warping network $\mathscr{W}$, and an OLAT prediction network $\mathscr{O}$. The feature network and the warping network output view-dependent feature maps, and the OLAT network takes per-texel features and a single lighting direction as input to predict the lighting-dependent colors at each texel. Finally we combine the colors under each light weighted by the lighting intensity to reproduce the texture. Please refer to Figure 5.3 for an illustration of our architecture.

Feature network. The feature network takes the latent vector, $\mathbf{z}$, and view direction, $\mathbf{v}$, as input and outputs a 64-channel feature map of size of 512×512:

$$\mathbf{C} = \mathscr{F}(\mathbf{z}, \mathbf{v}) \tag{5.4}$$

This feature map serves as a spatially varying encoding of expression and viewpoint across all lighting conditions.

Warping network. The warping network outputs a view-dependent warping field, $\mathbf{W} \in \mathbb{R}^{2\times 1024^2}$, which is applied to the feature map, $\mathbf{C}$, resulting in a warped feature map, $\tilde{\mathbf{C}} \in \mathbb{R}^{64\times 1024^2}$, of the same size as the texture:

$$\mathbf{W} = \mathscr{W}(\mathbf{z}, \mathbf{v}) \quad , \quad \tilde{\mathbf{C}}_t = \phi(\mathbf{C}, \mathbf{W}), \tag{5.5}$$

where ϕ denotes the warping operator, which performs bilinear interpolation at floating point coordinates. The warping field accounts for *texture sliding* as a result of view-dependent effects stemming from imperfect geometry, most noticeable around the mouth, eyes and hair, where

accurate geometry is difficult to estimate during mesh tracking. It is also used to upscale the lower resolution feature maps, whose size is constrained by memory limitations on modern GPU hardware.

OLAT network. Given the warped feature map, $\tilde{\mathbf{C}}$, $\mathscr{O}$ is applied to each texel independently, where it predicts the color of that texel under a given lighting direction. $\mathscr{O}$ is a multi-layer perceptron (MLP) that, for a texel k and a light b_i with position $\mathbf{l}_{b_i}$, takes as input $\tilde{\mathbf{C}}_k$, the 64-dimensional feature of $\tilde{\mathbf{C}}$ at texel k, as well as the direction of light with respect to the corresponding point on the face in 3D.

Different from previous works (e.g., [91]) which assume distant lighting and where all texels share the same lighting direction, we calculate the lighting direction of each texel using the light position and the corresponding position of the texel on the reconstructed geometry, $\hat{\mathbf{M}}$. This better models the setting in our light-stage, whose 1.1m radius results in some non-negligible foreshortening effects.

One of the most distinctive appearance change on faces is shadow by self-occlusion. While our late-conditioned model allows us to learn appearance change in a localized manner, we observe that it remains challenging for such a model to learn clear shadow boundary due to the lack of geometric information, resulting in noticeable artifacts. To alleviate this issue, we exploit the predicted geometry, $\hat{\mathbf{M}}$, to encode geometric relationship between a light source and a texel in the spirit of a shadow map [142] as an additional input to $\mathscr{O}$. Specifically, for a texel, k, and its corresponding 3D position, $\mathbf{p}_k$, we calculate the difference between the depth of $\mathbf{p}_k$ and its nearest occluder along the ray from the light to the texel in the light coordinate frame. With this, we arrive at the final form for our OLAT network:

$$\hat{\mathbf{T}}^{b_i}(k) = \mathscr{O}\left(\tilde{\mathbf{C}}_k, \mathbf{d}_k^{b_i}, s_k^{b_i}\right) \tag{5.6}$$

where $\mathbf{d}_k^{b_i}$ is the lighting direction of light b_i for texel k, and $s_k^{b_i}$ is the depth difference mentioned above. Applying the OLAT network to each texel gives us the full texture $\hat{\mathbf{T}}^{b_i}$ under the current

view direction and lighting, b_i.

Each frame of our training data is captured under multiple lights that we approximate by the weighted sum of textures generated for each light independently, using weights that reflect the intensity of each light. Given the preset lighting intensity γ^{b_i} for a light b_i, our final predicted texture is constructed as follows:

$$\hat{\mathbf{T}} = \sum_{i=1}^{n} \gamma^{b_i} \hat{\mathbf{T}}^{b_i}. \tag{5.7}$$

Training. Our loss function consists of four terms; a texture reconstruction loss ℓ_T, a geometry reconstruction loss ℓ_M, a regularizer loss on the warping field ℓ_W and a latent space regularizer ℓ_{KL}:

$$\mathscr{L}(\mathscr{E}_\ell, \mathscr{D}_\ell) = \sum_{v,t} \lambda_T \ell_T \; + \; \lambda_M \ell_M \; + \; \lambda_W \ell_W \; + \; \lambda_{\text{Z}} \ell_{\text{Z}}, \tag{5.8}$$

where $(\mathbf{v}, t)$ are the camera and frame indices over the dataset, and:

$$\ell_T = ||w \odot (\mathbf{T} - \hat{\mathbf{T}})||_2^2 \tag{5.9}$$

$$\ell_M = ||\mathbf{M} - \hat{\mathbf{M}}||_2^2 \tag{5.10}$$

$$\ell_W = ||\mathbf{W} - \mathbf{W_I}||_2^2 \tag{5.11}$$

$$\ell_Z = \text{KL}\big(\mathscr{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \;||\; \mathscr{N}(0, \mathbf{I})\big) \tag{5.12}$$

Here, w is a weight map that avoids penalizing self-occluded texels in the current view[2]. The term $\mathbf{W_I}$ is an identity warping field, and the regularizer loss ℓ_w prevents the warped texel positions from drifting too far from their original positions. The KL-divergence loss ℓ_{KL} with a standard normalization encourages a smooth latent space. In all our experiments we set the weights of each loss term as $\lambda_T = 1, \lambda_M = 0.1, \lambda_W = 10, \lambda_Z = 0.001$. We use the Adam optimizer [70] with a learning rate of 0.0005 for training. We train the networks on 4 Nvidia Tesla V100 GPUs with a batch size of 16 for about $300k$ iterations, which takes 4-5 days on average.

[2]We have omitted indexing the variables with $\mathbf{v}$ and t in our equations to reduce notation clutter, but they should be understood to correspond to unique values for every frame and viewpoint in the dataset.

Testing. Our model provides great flexibility and generalization for rendering under novel lighting conditions. We can feed in an arbitrary lighting direction for each texel as input to the OLAT network $\mathscr{O}$, and predict the texture under the desired lighting conditions. Therefore, our model supports the rendering of directional lighting (Figure 5.7), as well as *near-field* lighting (Figure 5.8), which is not previously possible using existing image-based portrait relighting methods [91, 129, 159][3]. For complex lighting conditions like environment maps, we can predict textures for every single pixel in the environment map, and linearly combine them to synthesize a face image in that environment. The model's runtime comprises: 24ms for shadow map calculation, 29ms for feature map generation, and 0.9ms for full texture decoding of a single lighting direction on a single Nvidia Tesla V100 GPU. Although feature map generation needs to be computed only once, the shadow map and texture decoding need to be performed for each light in the environment. So, although single light rendering using DRAM$_\ell$ can be relatively fast (i.e., $\sim$ 55ms), even a low-resolution (16×32)-environment map can take $\sim$ 18 seconds.

5.4.2 DRAM$_\mathcal{E}$: An Early-conditioned Model

Our late-conditioned model allows us to synthesize face images under novel expressions, viewpoints and lighting conditions. However, it is computationally expensive to evaluate for complex lighting conditions with many light sources. Unfortunately, most natural illuminations exhibit this property. Hence, they are typically modeled using an environment map, which is equivalent to having as many light sources as there are non-zero pixels in the map. Thus, this model is not suitable for interactive applications, such as VR, where real-time performance is necessary. In this section, we build on top of results from the late conditioned model described in the previous section to arrive at a formulation with similar accuracy, but that is an order of magnitude more efficient.

Data generation. We use DRAM$_\ell$ to generate face renderings under environment maps

[3]For nearfield lighting we employ a quadratic drop-off for lighting intensities used in the weighted sum in Equation 5.7.

captured from real indoor and outdoor scenes, and use the generated textures as ground truth to supervise the training of our early-conditioning model; $\text{DRAM}_{\varepsilon}$. For the set of environment maps to render, we use the same large-scale dataset used by Sun et al. [129], which contains 3094 high-resolution HDR environment illuminations including both indoor and outdoor scenes. We randomly select 2560 environment maps from the dataset for training and use the remaining 534 for testing.

We generate these synthetic lighting images for randomly sampled frames and viewpoints from our light stage capture. During rendering, we randomly select an environment map from the training dataset and apply a random rotation of $[0°, 360°]$ in longtitude and $[-30°, 30°]$ in latitude, followed by downsampling to a (16×32)_-sized lat-long environment map. The resized environment map is further normalized by dividing by its sum and multiplying by a constant $\alpha \in [6, 12]$. We denote the environment map as $\mathbf{e}$. DRAM_{ℓ} is applied to predict the textures for each lighting direction, which correspond to individual pixels in the environment map, and perform the weighted sum in Equation 5.7 to produce the final texture $\mathbf{T}^{e}$. In total, we generate $1.2M \sim 1.8M$ textures for training each subject in our dataset. In addition to environment map renderings, we also augment our training data by rendering the captured subject under $1-5$ lights randomly selected from the 460 lights. During training, we project the selected lights onto an environment map of 16×32 and use them as input to our network to predict the corresponding textures.

Network architecture. Our early-conditioned model exhibits a similar CVAE architecture as its late-conditioned counterpart, comprising an encoder, $\mathscr{E}_{\varepsilon}$, and a decoder, $\mathscr{D}_{\varepsilon}$. The encoder, $\mathscr{E}_{\varepsilon}$, shares the same architecture and input as $\mathscr{E}_{\ell}$, and outputs a latent vector, $\mathbf{z}$. The decoder also consists of two branches; a geometry decoder, $\mathscr{G}_{\varepsilon}$, with the same architecture as $\mathscr{G}_{\ell}$, and a texture decoder, $\mathscr{T}_{\varepsilon}$, that predicts a texture under the given environment map.

As shown in Figure 5.4, a naïve architecture for the texture decoder would be an extension of [85], where the vectorized environment map is concatenated with the latent vector, $\mathbf{z}$, and view

direction, $\mathbf{v}$, and fed into to a single deconvolutional network to output the predicted texture, $\hat{\mathbf{T}}^e$. As this network architecture is designed for speed, it lacks the capacity to accurately reconstruct data that spans a large number of different environment maps. To do this, a straight-forward approach would be to increase the channel size in the hidden layers of the network. However, as we will show in §5.6.2, a considerable increase is required to achieve reasonable accuracy, which diminishes the model's efficiency, making it unsuitable for real-time applications.

DRAM$_\varepsilon$ takes inspiration from recent works on hyper-networks [50], and consists of two networks: a weights network, $\mathscr{H}$, that takes the environment map as input and predicts the weights for a second network, $\mathscr{T}_\varepsilon$, that takes the efficient form used in [85], and produces a view, lighting and expression dependent texture:

$$\Theta \leftarrow \mathscr{H}(\mathbf{e}) \quad , \quad \hat{\mathbf{T}}^e = \mathscr{T}_\varepsilon(\mathbf{z}|\mathbf{v},\Theta). \tag{5.13}$$

Θ denotes the weights of $\mathscr{T}_\varepsilon$ that consists of 8 transposed convolution layers. For each layer, we use a small weights network that consists of 5 fully connected layers to predict the convolutional kernel weights and biases. Similar to the late-conditioned decoder, a warping field is employed on the output of the texture decoder to give us the final texture. The hyper-network architecture specializes the texture network to a specific lighting condition, which we find to be effective in improving reconstruction performance without substantially increasing computational cost, as shown in Figure 5.14 and Table 5.5.

Training. We use all the same settings for training DRAM$_\varepsilon$ as we did for DRAM$_\ell$. For the same number of iterations, a model can be trained within $3-4$ days on average. The trained model can synthesize face images lit by environment maps within 13ms ($\sim$75 frames per second), making it suitable for interactive applications, including demanding real-time applications such as VR.

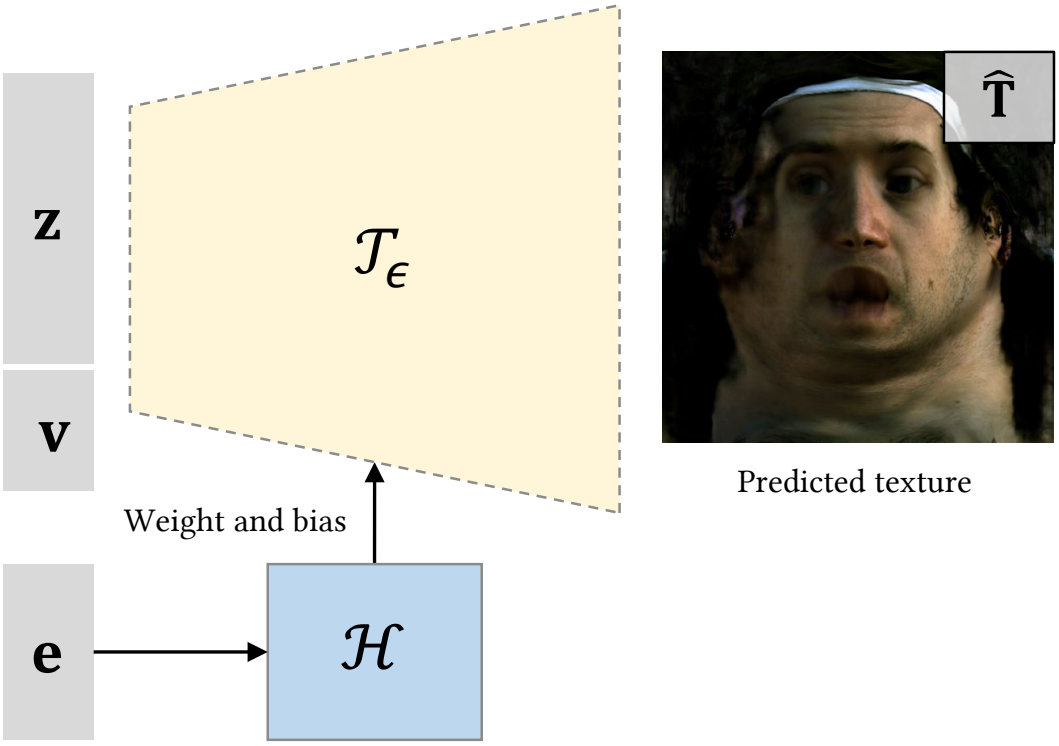

Figure 5.4. We apply a hyper-network architecture for our early-conditioned model, where we use a separate network $\mathscr{H}$ that takes the current environment map as input to predict the weight and bias of the texture decoder $\mathscr{T}_\varepsilon$.

5.5 Animating Relightable Avatars

The trained early-conditioned decoder $\mathscr{D}_\varepsilon$ can efficiently generate novel outputs with respect to its three inputs: expression, viewpoints, and lighting. The disentanglement of these factors in the model are important for animation, because the images coming from driving sensors can have completely unrelated viewpoints and lighting to the decoded avatar. For example, in the VR telepresence system of [138], the driving signal comes from headset-mounted IR cameras that observe facial expressions of a person wearing the headset in an arbitrary room, while being lit by headset-mounted IR lights, whereas the avatar that person is driving needs to be relit in accordance with the virtual scene, which might be arbitrarily different from where the person really is. The only factor that is desirable to match between the sensor images and the avatar, is the facial expression.

In this work, we utilize the method in [116], which finds correspondences between input headset images and expression codes $\mathbf{z}$ of DRAM_ℓ through analysis-by-synthesis. We similarly learn a regressor that encodes multi-view headset images into $\mathbf{z}$ and a relative pose between the headset and avatar, jointly with a style transformer that accommodates for domain differences between the headset images and the rendered avatar. An important difference here, is that we assume the lighting variation in the sensor images is small enough so that we can fix the lighting input, $\mathbf{e}$, at a constant uniform lighting. Any difference in lighting between the domains is handled by the style transformer. While this assumption holds in many cases, as shown in Section 5.6, an interesting future direction is to leverage our model's relighting capability and jointly optimize the model's lighting so that there is less reliance on the style transfer module, which can introduce semantic shifts during optimization.

5.6 Results

In this section we provide qualitative and quantitative evaluations of different components of our method including both the late-conditioned model (Section 5.6.1) and the early-conditioned model (Section 5.6.2). We perform ablation studies on each model and validate our design choices. We show relighting results with our models under novel lighting conditions, viewpoints, and expressions. We also demonstrate our relighting results animated by VR headset mounted cameras (Section 5.6.3).

5.6.1 Evaluation of late-conditioning model

As discussed in Section 5.2, none of the previous works support both free-viewpoint relighting and animations, as in our method. The work that is closest to ours is Meka et al. [92]. However, their model is not animatable and requires color gradient illuminations as input. Therefore, in this section we focus on showing our qualitative results and validating different design choices in our system.

Figure 5.5. We show comparisons between the predicted OLAT images under novel viewpoints and expressions with our late-conditioned model and the ground truth. Our model is able to reconstruct the OLAT images accurately, even though it is trained only on group-light captures. This enables us to synthesize accurate renderings under novel lighting conditions such as environment lighting by combining multiple OLAT predictions.

Qualitative results. We first compare our renderings to ground truth captures under novel viewpoints, expressions, and lighting conditions. To achieve this, we evaluate our model on a separate sequence of images captured using a similar acquisition setup as described in Section 5.3 except that each frame in this sequence is captured under a single light. We make the comparison on images captured at a set of 4 validation viewpoints that are not used in training. As shown in Figure 5.5, although our model is trained on images captured under group-light patterns and has never seen OLAT captures during training, our network can successfully reconstruct high-fidelity OLAT images that closely approximate the ground truth captures in terms of shadows, specularities, and texture details. This demonstrates that our proposed model can not only generalize to novel expressions and viewpoints, but also effectively super-resolve the group-light captures and increase the resolution of the lighting.

Figure 5.6 shows renderings with our model under novel directional lights. By combining the renderings under each pixel lighting of an environment map, our model can also achieve

Figure 5.6. Late-conditioned model: rendering under novel *directional lights*.

Figure 5.7. Renderings under environment maps with our late-conditioned model. Our model is able to faithfully recover complex shading effects including specularities and shadows.

photorealistic renderings under environment lighting. Figure 5.7 shows environment map renderings with our model under both outdoor and indoor environment maps. Our model can faithfully recover the glints on the forehead and the specularities on the face.

Figure 5.8 shows our rendering results under near-field lighting. We make use of our reconstructed geometry output by the geometry decoder to calculate the lighting direction of each texel. Since our OLAT prediction network $\mathscr{O}$ is applied on each individual texel, our model can predict the OLAT renderings with a single inference. In comparison, previous methods [155, 129, 91] do not reconstruct the geometry and therefore fail to support near-field

Figure 5.8. Nearfield relighting with our late-conditioned model. Our late-conditioned model can take different lighting directions for each texel and predict their colors, which enables us to achieve efficient nearfield renderings making use of the geometry reconstructed with our geometry decoder.

lighting. While some other methods [159, 92] build on estimated geometry, their network can only take a single lighting direction as input at each time. To predict near-field rendering, separate evaluations of their model for the lighting direction of each individual pixel would be required, which is very time-consuming. In contrast, our model provides greater flexibility and more efficient near-field renderings. For more results, please refer to the supplementary video of the original paper.

Evaluation of design choices. To validate our different design choices, we evaluate our models on testing sequences and compare them to the ground truth. We consider image-space error metrics including mean-squared error (MSE) and structural similarity index (SSIM). Considering the fact that the ground truth OLAT images with our models may have different lighting intensity than the predictions, and there are potential color mismatches due to different camera calibrations, we optimize a matrix $\mathbf{Q} \in \mathbb{R}^{3\times3}$ to align our predicted image $\hat{\mathbf{I}}$ to the ground

Ground truth Ours w/o depth difference

Figure 5.9. We make use of the shadow mapping technique to feed self-shadowing information to the network. We can see that without the depth differences, the rendering results suffer from jagged boundaries at shadows. In contrast, our full model reproduces more accurate shadows.

truth $\mathbf{I}$:

$$\mathbf{Q} = \arg\min_{\mathbf{Q}} ||\mathbf{Q}\hat{\mathbf{I}} - \mathbf{I}||_2^2 \tag{5.14}$$

Then we calculate all error metrics between $\mathbf{Q}\hat{\mathbf{I}}$ and $\mathbf{I}$.

Figure 5.10. We show visual comparisons between renderings on novel expressions with our late-conditioned model trained on static captures and dynamic captures. Static captures cover many fewer facial expressions than dynamic captures within the same number of frames, thus resulting in poor generalization to novel expressions.

In Table 5.2, we perform an ablation study to show the effectiveness of applying depth differences as input to our OLAT network. From the result we can conclude that it helps improve the accuracy of the model. We provide additional qualitative comparisons in Figure 5.9. As we can see from the figure, without including the depth difference information, the network predicts shadows with incorrect shapes and jagged boundaries, especially for long-range shadows on the

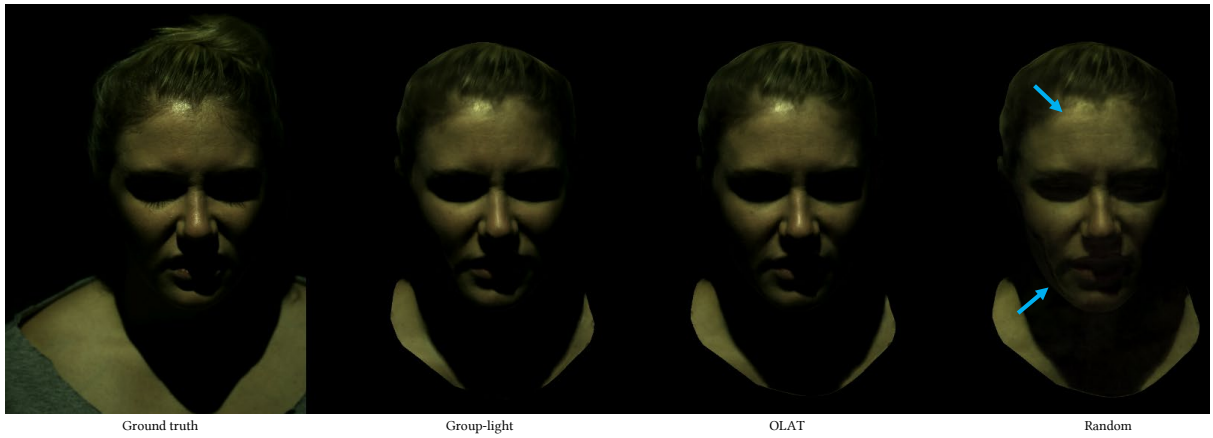

Figure 5.11. Late-conditioned model: a visual comparison of different *capture lighting patterns*. The leftmost image shows a ground truth image under an "OLAT" single point-light illumination. We reconstruct this using a model trained on 5 spatially clustered lights ("Group-light"), OLAT, and 5 spatially random lights ("Random"). Both "Group-light" and "random" can use shorter camera exposures than OLAT to achieve similar camera intensities, but only "Group-light" recovers comparable details to OLAT.

Table 5.2. We evaluate the effectiveness of using depth differences as input to the OLAT network in our late-conditioned model on two subjects. Subject 1 and Subject 2 correspond to two subjects in Figure 5.5 respectively, The results on both subjects show that involving the occlusion information helps improve the accuracy of our model.

	Subject 1		Subject 2	
	MSE ($\times 10^{-4}$)	SSIM	MSE ($\times 10^{-4}$)	SSIM
Our full model	**6.4377**	**0.9363**	**2.9843**	**0.9469**
w/o depth difference	6.5115	0.9344	3.0562	0.9464

neck. In contrast, our full model with depth differences produces more accurate shadows.

Effect of spatial ight pattern. In Table 5.3, we make a comparison between different lighting bases for our time-multiplexed lighting. We train our late-conditioned models on captures under different lighting bases including OLAT, Random, and Group-5. We make the comparisons by predicting the OLAT images under novel expressions and viewpoints and calculate the error between the predictions and their corresponding ground-truth OLAT captures. A visual comparison is also shown in Figure 5.11. From the results we can see that Group-5 captures lead to better reconstruction accuracy than *r*andom lighting patterns. Compared to

Table 5.3. We compare the performance of our models trained on captures under different basis lighting patterns. We do the evaluation by comparing the predicted OLAT images under novel viewpoints and expressions to their corresponding ground-truth. From the result we can see that the performance of Group-5 capture is much better than random light patterns. The Group-5 capture is even better than the OLAT captures on 3 out of 4 metrics although it has never seen OLAT images in training. Note that this evaluation is done on a different testing sequence from that used in Table 5.2.

	Subject 1		Subject 2	
	MSE ($\times 10^{-4}$)	SSIM	MSE ($\times 10^{-4}$)	SSIM
OLAT	6.5071	0.9349	**3.4949**	0.9415
Random	9.9668	0.8842	4.1159	0.9230
Group-5	**6.3761**	**0.9368**	3.6791	**0.9433**

Table 5.4. Late-conditioned model: comparison between static captures and dynamic captures. Within the same number of frames, dynamic captures can cover more facial expressions and lead to better generalization to novel expressions, thus achieving higher accuracy.

	Subject 1		Subject 2	
	MSE ($\times 10^{-4}$)	SSIM	MSE ($\times 10^{-4}$)	SSIM
Static capture	7.5862	0.9301	3.6335	0.9429
Dynamic capture	**6.4377**	**0.9363**	**2.9843**	**0.9469**

OLAT, Group-5 achieves very similar performance despite the model never having seen OLAT images during training, and using an evaluation design that is favorable to OLAT captures. Grouped light captures have reduced single-light maximum power requirements compared to OLAT to overcome the noise floor of the cameras, or, equivalently, support capturing with shorter exposure times (which has implications for perceptual discomfort [139]). This result indicates that grouping lights in spatial clusters is an attractive option for power or exposure constrained settings, with results almost indistinguishable for groups with diameter twice the size of the single-light spatial sampling distance on average.

Effect of capture script content. We also compare a dynamic capture script with a static expression capture script of roughly the same total duration. For static captures, the subject is asked to remain still during each elicited expression, while a full cycle of the light patterns is captured. Each individual expression is therefore fully sampled along all spatial lighting

Table 5.5. We compare our early-conditioned model with hyper-networks against baseline models. Compared to the naïve model that applies a single decoder network with fixed weights, our hyper-network is able to achieve much better accuracy with similar computational cost. Our model is even comparable to the naïve decoder model with twice as many feature channels at each layer, which has a much larger computational cost. In addition, from the figure we can see that while the model trained on group-light captures only can predict accurate group-light renderings, it fails to generalize to novel environment maps, which demonstrates the necessity of our two-stage student-teacher framework.

	Test on environment renderings				Test on group-light captures				MACs ($\times 10^9$)
	Subject 1		Subject 2		Subject 1		Subject 2		
	MSE($\times 10^{-4}$)	SSIM	MSE ($\times 10^{-4}$)	SSIM	MSE($\times 10^{-4}$)	SSIM	MSE($\times 10^{-4}$)	SSIM	
Single decoder	11.396	0.9815	3.6423	0.9885	7.5256	0.9833	3.7622	**0.9937**	1.44
Single decoder ($\times 2$ features)	8.7349	0.9862	2.6697	0.9909	7.2872	0.9838	3.5373	0.9938	5.53
Ours (train on group-light)	108.93	0.9277	61.221	0.9417	**6.2536**	**0.9846**	**3.0830**	**0.9937**	1.50
Ours	**7.3878**	**0.9882**	**2.5309**	**0.9914**	7.4345	0.9838	3.5846	0.9938	1.50

directions. Conversely, for the dynamic capture, the subject is asked to move naturally, and the allotted capture time is used to elicit more varied expressions and poses. Instantaneous expressions are therefore sampled very sparsely along lighting directions, but a more diverse set of facial expressions is sampled, and we rely on amortized inference during model building to span the combined space of expressions and lighting directions. We capture roughly the same number of frames for these two kinds of captures. From the result in Table 5.4 we can see that our dynamic captures produce better results than static captures. Specifically, within the same number of frames, static captures cover a much smaller set of expressions than our dynamic captures. Therefore, as we can see in Figure 5.10, the model trained on static captures does not generalize well to novel expressions. In contrast, our dynamic captures provide us a more efficient way to capture the subject under a large number of expressions.

5.6.2 Evaluation of early-conditioned models

Qualitative results. In Figure 5.12, we show the rendering results with our early-conditioned model under novel environment maps, expressions and viewpoints. Because we are using the late-conditioned model to supervise the training of the early-conditioned model, we regard the renderings with the late-conditioned model as the ground truth. From the results, we can see that our early-conditioned model can generate photorealistic results with accurate texture

Figure 5.12. Our *early-conditioned* model is able to generate renderings under novel environment maps that have the same quality as those generated by the *late-conditioned* model. Compared to the time of around 18 seconds required by the late-conditioned model, our early-conditioned model is much more efficient and can generate environment map renderings in real time.

Figure 5.13. We compare our early-conditioned model to the state-of-the-art single image portrait relighting method of Sun et al. [130]. From the results we can see that the method of Sun et al. fails to recover accurate specularities on the face and eyes, produces softer shadows, and predicts incorrect colors. In contrast, our method can achieve much more photorealistic relighitng.

details and shading effects that closely resemble its corresponding ground truth. Such results demonstrate that by extensively sampling the natural illuminations and generating renderings as training data, our early-conditioned model is able to achieve good generalization to novel lighting conditions.

We also compare to the state-of-the-art single image portrait relighting work of Sun et al. [129]. To achieve this, we directly feed the ground truth fully-lit captured image as input to their method and compare their renderings to the predictions of our early-conditioned model. We use the code and model provided by Sun et al. to generate the results. From the results in Fig. 5.13, we can see that the method of Sun et al. fails to predict faithful shading effects such as

Figure 5.14. We compare our early-conditioned model with hyper-networks against baseline models with a single decoder. Our hyper-network generates results of higher quality that better match the ground truth compared to baseline models. We also compare against a model that has the same architecture as ours but is trained on group-light captures only, and the result shows that such a model fails to generalize to novel lighting conditions. Instead, our models that are trained on environment map renderings with our late-conditioned model achieve better generalization, which demonstrates the effectiveness and necessity of our two-stage student-teacher framework.

specularities, and generates overly flat renderings. In contrast, our method produces renderings with higher fidelity.

Evaluation of design choices. To validate the design choices of our early-conditioned model, we evaluate our model and the comparison models on a separate testing sequence with novel expressions. We generate renderings with the models at a set of 4 novel viewpoints under environment maps randomly chosen from the testing dataset. The corresponding ground truth renderings are generated with our late-conditioned model. We also compare the renderings of different models on a testing sequence under group-light patterns to the ground truth group-light captures. To achieve this, we project the group-light patterns to environment maps and use them as input to the models. We apply the same error metrics as used in Section 5.6.1, and report the scores in Table 5.5. We also report the computational cost of the texture module by calculating the number of multiply-accumulate operations (MACs) for a single inference. We also show a

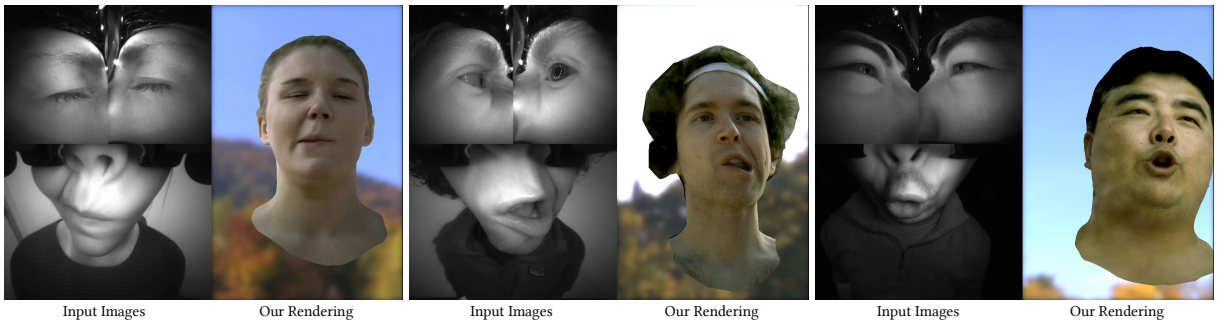

Figure 5.15. Our early-conditioned model under novel environment maps animated by VR headset mounted cameras. Our model is able to faithfully reproduce the expressions in the headset captures while achieving photorealistic relighting simultaneously.

visual comparison between our model and the baseline models in Figure 5.14.

From the results we can see that the naïve decoder with fixed weights has low accuracy and generates incorrect colors on faces. In comparison, our hyper-network architecture produces more accurate renderings while maintaining a comparable computational cost. We also compare against a baseline model with twice the number of feature channels, and our hyper-network is able to achieve better performance with a much smaller computational cost.

Instead of training on environment map renderings generated with our late-conditioned model, we also compare against a model that is only trained on our group-light captures by projecting the group-light onto environment maps and training the same hyper-network model on this dataset. From the table we can see that while it can produce renderings with the highest accuracy on a testing set of group-light sequences, it produces the lowest accuracy on the environment renderings. Training only on the group-light captures makes the network overfit to the training lighting patterns and fail to generalize to novel environment lighting (see Figure 5.14, right). In contrast, by training on renderings with our late-conditioned model under extensively sampled natural illuminations, our hyper-network can generalize to novel environment illuminations, which demonstrates the necessity and effectiveness of our two-stage framework.

5.6.3 Animation from headset mounted cameras

The advantage of our relightable appearance model over previous works is the good disentanglement of its inputs: facial expression, viewpoints, and lighting. This makes the model animatable, and can be driven by sensors such as video captured by VR headset mounted cameras. In Figure 5.15, we show our early-conditioned model relighted in different environment, given the latent values $\mathbf{z}$ we extracted from the given headset images, using the method described in Section 5.5. The renderings of our model faithfully reproduce the facial expressions from the headset images, while photorealistically relit under novel lighting conditions. For more results, please refer to the supplementary video of the original paper.

5.6.4 Limitations

While our models produce photorealistic relightable avatars, several limitations remain: (1) Most notably, in regions where the tracked mesh is inaccurate or lacks sufficient geometric detail, such as the hair outline, textured mesh rendering produces jarring border artifacts instead of blending into the environment background. This could potentially be addressed via volumetric neural rendering approaches with the capacity to produce translucency. (2) Similarly, we notice some blurring in regions where the mesh geometry does not accurately track the surface, such as the mouth and eyes. Using specialized geometric models for these regions (e.g., [12, 145]) would greatly improve registration accuracy and therefore reduce the capacity required to model their appearance in texture space. (3) Our models are limited by the acquisition hardware and lighting rig used to capture the training data. Due to the use of low-dynamic range 8-bit images, we notice decreased quality and color shifts in very dark regions, likely due to poor signal to noise ratio. Similarly, our model fails to reconstruct lighting directions that are very far from those that can be elicited by the light stage (e.g., lighting directly below the participant). High bit-depth HDR imaging and more complex light stage setups could improve results in these cases. (4) Finally, we have presented an efficient model for rendering animatable and relightable avatars

in realtime in environmental illumination, but designing an efficient model that can render both nearfield and farfield illumination remains an open problem.

5.7 Conclusion

We presented Deep Relightable Appearance Models, a novel two-stage framework to achieve photo-realistic relighting of animatable face avatars. Our approach produces, for the first time, a photo-realistic face avatar that can be driven and rendered in real-time under various new illuminations. The experiments demonstrate that our late-conditioned model achieves high generalization across a wide-range of illuminations including natural indoor/outdoor illuminations, nearfield lighting, and distant directional lighting, despite being trained only with grouped point-light captures. This is possible due to the explicit modeling of the additive property of light transport and the late-stage fusion of light information in our network architecture. We further examined the effects of different light patterns and captured scripts, and show the efficacy of dynamic capture and spatial grouping of light sources. This allows us to render high-quality synthetic images under different illuminations to generate an augmented training set for training efficient models. We also presented a hyper-network architecture for early-conditioned relightable models, which is highly efficient to run in real-time while showing comparable fidelity to a higher-capacity baseline. We believe that our two-stage framework is general and applicable to many different relighting problems and real-time applications, including volumetric rendering, and building cross-identity face models, which can be addressed in future work.

This chapter is based on the material as submitted to ACM Transactions on Graphics, 2021 (“Deep Relightable Appearance Models for Animatable Faces”, Sai Bi, Stephen Lombardi, Shunsuke Saito, Tomas Simon, Shih-En Wei, Kevyn McPhail, Ravi Ramamoorthi, Yaser Sheikh, Jason Saragih). The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusion and Future Work

In conclusion, in this dissertation we have presented four works on acquiring the appearance of real-world objects and scenes. We apply appropriate representations for scene geometry and reflectance to support functionalities such as changing viewpoints, relighting and dynamic animations. For diffuse objects, in Chapter 2 we have introduced a novel method to recover high-quality texture maps from inaccurate RGB-D reconstructions. Our method can effectively handle the misalignment of input images caused by inaccurate camera poses, corrupted geometries and optical distortions, and generate aligned images as a result of our optimization framework. It enables us to produce compelling textures for the RGB-D scans without blurry or ghosting artifacts, which greatly improves the quality of 3D reconstructions with consumer RGB-D cameras. For non-Lambertian objects, we first introduce a learning-based method to reconstruct high-quality geometry and per-vertex BRDFs in Chapter 3. Different from previous works that require dense input images that are time-consuming to capture, our method takes a sparse set of six images captured under collocated camera and light as input. We apply data priors to solve this challenging problem and propose novel networks for geometry and SVBRDF estimation by aggregating multi-view cues. Our method enables reconstruction of objects with complex non-Lambertian reflectance with a simple acquisition setup. In Chapter 4, we further introduce a novel volumetric representation to encode the geometry and reflectance properties of the scene. Such a novel volumetric representation can accurately model objects

and scenes with complex geometry such as thin structures and heavy occlusions, and supports free-viewpoint relighting under arbitrary lighting conditions. We have proposed an end-to-end method based on differentiable ray marching to learn such a representation from unstructured mobile phone captures under a flash light. Finally, in Chapter 5 we go beyond static scenes and introduce a relightable appearance model for animatable faces where we directly apply neural networks to regress the geometry and textures under the desired expressions, view directions and lighting conditions. We propose novel methods to train such networks from efficient dynamic captures under time-multiplexed lighting. Our method faithfully creates an animatable and relightable avatar that can be driven by VR headset captures, enabling many applications such as telepresense.

We demonstrate that our methods can effectively reproduce the appearance of real-world objects and scenes by recovering their geometry and material properties from the captured images. Compared to manually creating 3D content with complicated software that requires time-consuming efforts and special expertise, this dissertation takes one step further towards practical and faithful appearance acquisition, which empowers novice users to create digital 3D content by simply capturing a set of images. We believe that efficient and accurate methods for digital 3D content creation is the key to the future of many applications such as VR and AR, and many interesting topics remain to be explored. Here we discuss some future directions in this field:

Unconstrained environment. In Chapter 3 and 4 we have introduced methods to achieve relightable reconstructions of objects using only collocated camera and light, which enables mobile phone captures with a flash light. While such an acquisition setup is less expensive than those used previous methods [9, 53, 132, 163], it still requires the captures to be conducted in a dark room, which limits the practicability of these methods. To make appearance acquisition more practical, ideally we would like to enable casual mobile phone captures under mildly constrained or even unconstrained environments such as indoor scenes. Unknown environment

illumination leads to more ambiguities in material estimation, and the introduction of data priors as done in [83] would alleviate the problem and help achieve better material estimation.

Complex light transport effects. In this dissertation, we mainly focus on recovering the appearance of solid opaque objects, and only direct illumination is considered in the imaging formation models. Real-world objects and scenes have more complex materials such as transparency and translucency, and more complex light transport effects such as refractions, inter-reflections and subsurface scattering. Accurate appearance acquisition with the existence of these complex light transport effects remains challenging. Recently the development of differentiable rendering using Monte Carlo estimators [158] allows us to calculate the derivative with respect to arbitrary scene parameters in the presence of complex light transport effects. It would to be interesting to apply this framework for the task of appearance acquisition.

Neural representations for large-scale scenes. In addition to acquiring the appearance of real-world objects, another direction would be developing novel representations to reproduce the appearance of large-scale scenes. Traditional methods [3, 127] rely on 3D reconstructions based on multi-view stereo or depth sensors. Such methods fail to handle complex scenes with thin structures, heavy occlusions and result in reconstructions with holes. They cannot accurately reproduce the view-dependent effects of non-Lambertian surfaces, which greatly limits the photorealism of the rendered images. Recently neural-rendering based methods [94, 95] have shown great progress in generating highly photorealistic images. It's exciting to scale these methods to large-scale scenes so that we can freely navigate them in VR.

Neural representations from sparse images. We have introduced novel neural representations to recover the appearance of real-world objects. In Chapter 4, we introduced a novel neural volumetric representation for joint view synthesis and relighting. In Chapter 5 we train neural networks to regress the geometry and textures under novel expressions, viewpoints and lighting conditions. While such a model can faithfully reproduce the appearance of the captured object, the model needs to be trained from scratch for each object with abundant input images,

which is highly time-consuming. In the future, it would be desirable to research into methods that can effectively generalize to novel scenes with a sparse set of images or even a single image as input.

Appendix A

Appendix for Chapter 1

In this Appendix, we discuss the derivation of Eq. 2.8, which computes the targets that minimize the second term of Eq. 2.5. To start, we rewrite E_2 as:

$$E_2 = \frac{1}{N} \sum_{k=1}^{N} \sum_{\mathbf{x}_k} \sum_{j=1}^{N} w_j(\mathbf{x}_{k\to j}) \left(T_j(\mathbf{x}_{k\to j}) - M_k(\mathbf{x}_k) \right)^2. \tag{A.1}$$

To compute the optimum targets, we first need to differentiate the error with respect to each target as:

$$\frac{\partial E_2}{\partial T_i(\mathbf{x}_i)} = \frac{\partial \sum_{k=1}^{N} \sum_{\mathbf{x}_k} w_i(\mathbf{x}_{k\to i}) \left(T_i(\mathbf{x}_{k\to i}) - M_k(\mathbf{x}_k) \right)^2}{\partial T_i(\mathbf{x}_i)}, \tag{A.2}$$

where we remove the normalization factor, since it does not affect the optimum result. Moreover, since we differentiate with respect to the i^{th} target, we set $j = i$. Here, for each k, the summation is over all pixels of image k. Since we take the derivative with respect to the i^{th} target, we should backproject each term from k to i. By ignoring the effect of interpolation in the projection, we have:

$$\frac{\partial E_2}{\partial T_i(\mathbf{x}_i)} = \frac{\partial \sum_{k=1}^{N} \sum_{\mathbf{x}_i} w_i(\mathbf{x}_i) \left(T_i(\mathbf{x}_i) - M_k(\mathbf{x}_{i\to k}) \right)^2}{\partial T_i(\mathbf{x}_i)}, \tag{A.3}$$

where we used the fact that $\mathbf{x}_{i\to k\to i} = \mathbf{x}_i$. By taking the derivative in the above equation and

setting it equal to zero, T_i's can be calculated as defined in Eq. 2.8. Note that, since the derivative is with respect to a single pixel of the target image $\mathbf{x}_i$, we remove the summation over all pixels before taking the derivative.

Appendix B

Appendix for Chapter 2

B.1 BRDF Model

We use a simplified version of the Disney BRDF model [18] proposed by Karis et al. [67]. Let A, N, R, S be the diffuse albedo, normal, roughness and specular albedo respectively, L and V be the light and view direction, and $H = \frac{V+L}{2}$ be their half vector. Our BRDF model is defined as:

$$f(A,N,R,L,V) = \frac{A}{\pi} + \frac{D(H,R)F(V,H,S)G(L,V,H,R)}{4(N\cdot L)(N\cdot V)} \tag{B.1}$$

where $D(H,R)$, $F(V,H,S)$ and $G(L,V,H,R)$ are the *normal distribution*, *fresnel* and *geometric terms* respectively. These terms are defined as follows:

$$\begin{aligned}
D(H,R) &= \frac{\alpha^2}{\pi\left[(N\cdot H)^2(\alpha^2-1)+1\right]^2} \\
\alpha &= R^2 \\
F(V,H,S) &= S+(1-S)2^{-[5.55473(V\cdot H)+6.8316](V\cdot H)} \\
G(L,V,R) &= G_1(V,N)G_1(L,N) \\
G_1(V,N) &= \frac{N\cdot V}{(N\cdot V)(1-k)+k} \\
G_1(L,N) &= \frac{N\cdot L}{(N\cdot L)(1-k)+k} \\
k &= \frac{(R+1)^2}{8}
\end{aligned}$$

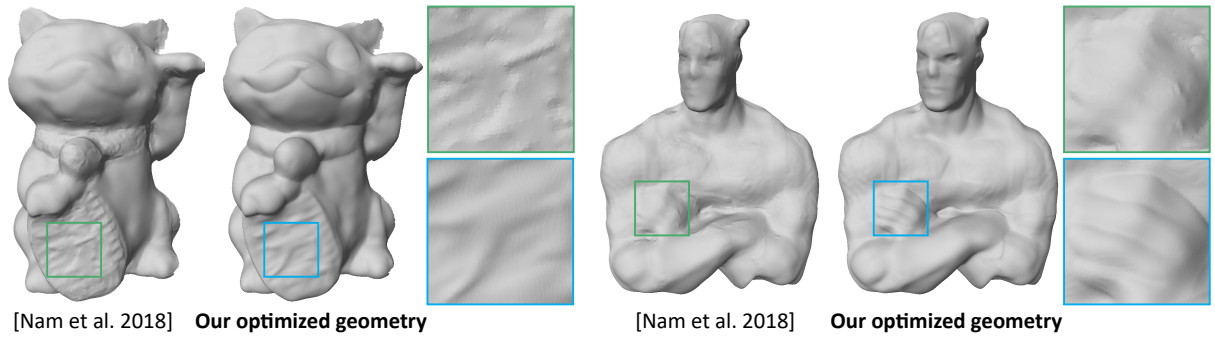

Figure B.1. Comparison with Nam et al. [97] on geometry optimization. Our results have more fine-grained details and fewer artifacts.

Network input	Diffuse	Normal	Roughness	Specular
$I_{i\leftarrow j}$	0.0081	0.0456	0.0379	0.0098
$I_i, I_{i\leftarrow j}$	0.0071	0.0363	0.0304	0.0109
$I_i, I_{i\leftarrow j}, Z_{i\leftarrow j}, Z^*_{i\leftarrow j}$	0.0063	0.0321	0.0306	0.0098
$I_i, I_{i\leftarrow j}, L_i, L_{i\leftarrow j}$	**0.0061**	**0.0304**	0.0299	0.0093
Ours full	**0.0061**	**0.0304**	**0.0275**	**0.0086**

Table B.1. Quantitative comparisons between networks trained with different inputs on the synthetic test set.

B.2 Comparison on Geometry Reconstruction

In Fig. reffig:comp-nam-geo of Chapter 3, we compare our optimized geometry against the optimized result from Nam et al. [97] that uses the same initial geometry as ours. We show additional comparisons on real data in Fig. B.1. Similar to the comparison in the paper, our optimized geometry is of much higher quality than Nam et al. with more fine-grained details and fewer artifacts.

B.3 Additional Ablation Study

In this section, we demonstrate additional experiments to justify the design choices in our pipeline, including input variants of the SVBRDF estimation network, non-rigid warping and per-vertex refinement.

Network inputs. Our SVBRDF network considers the input image (I_i), the warped

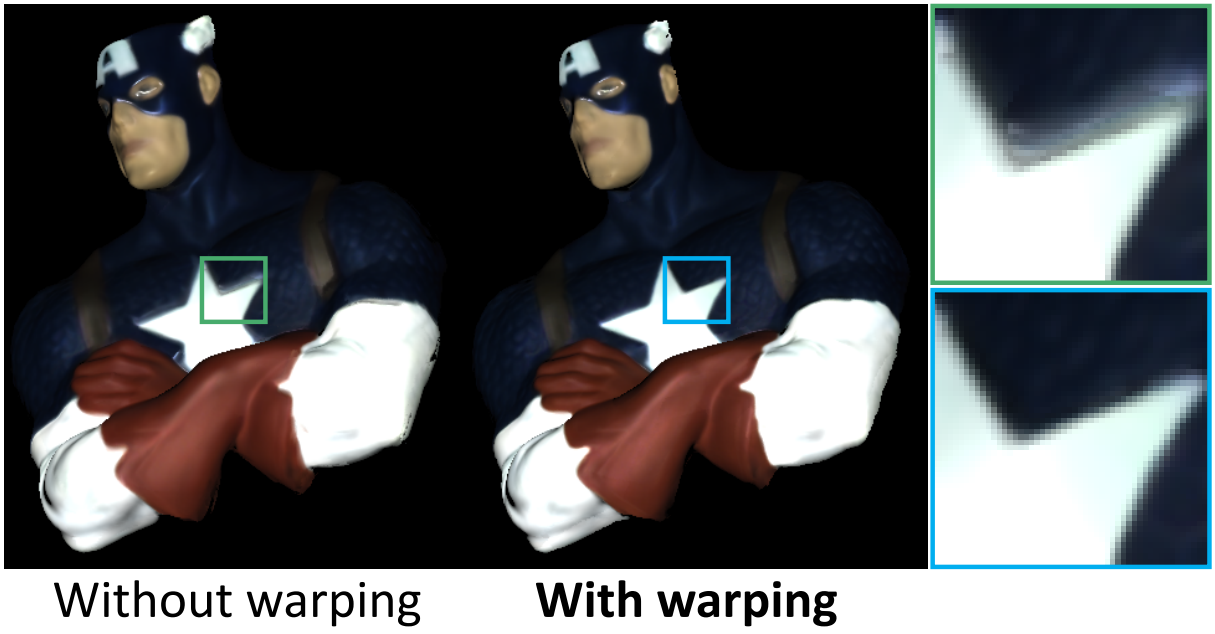

Figure B.2. Comparison between optimizations with and without per-view warping. Our method with warping removes the ghosting artifacts around the edges.

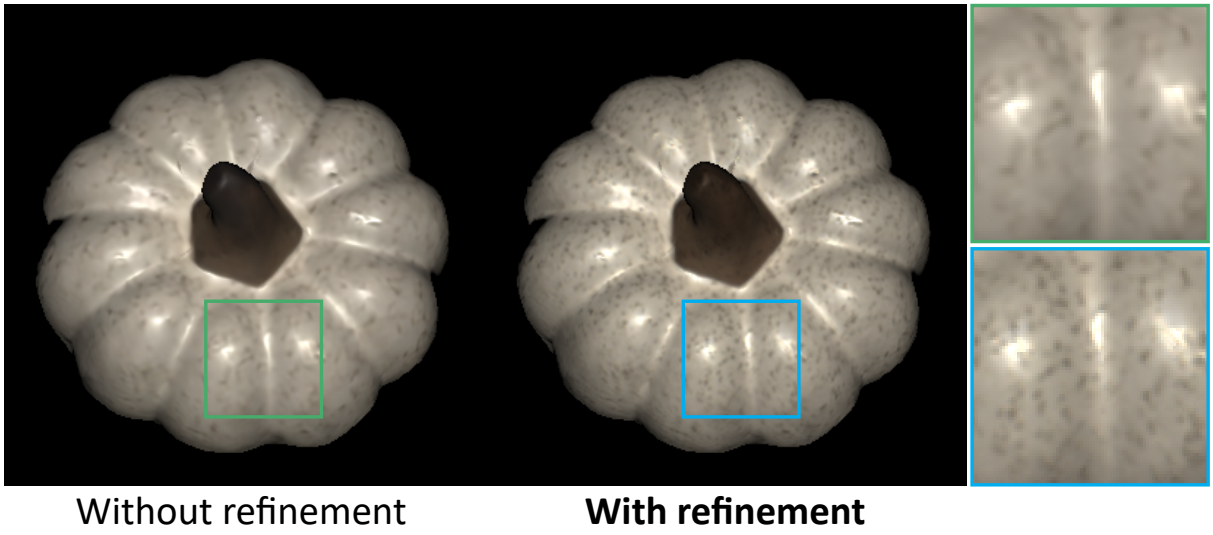

Figure B.3. Comparison on results with and without per-vertex refinement. With the refinement, our method is able to recover high-frequency details such as the spots on the object.

images ($I_{i\leftarrow j}$), the light/viewing (which are collocated) direction maps (L_i and $L_{i\leftarrow j}$), and the depth maps ($Z_{i\leftarrow j}$ and $Z^*_{i\leftarrow j}$) as inputs (please refer to Sec. 3.3.2 in the paper for details of these input components). We verify the effectiveness of using these inputs by training and comparing multiple networks with different subsets of the inputs. In particular, we compare our full model

against a network that uses only the warped image $I_{i\leftarrow j}$, a network that considers both $I_{i\leftarrow j}$ and the reference image I_i, a network that uses the reference image, warped image and the depth, and a network that uses the reference image, warped image, and the viewing directions. Table. B.1 shows the quantitative comparisons between these networks on the synthetic testing set. The network using a pair of images (I_i, $I_{i\leftarrow j}$) improves the accuracy for most of the terms over the one that uses only the warped image ($I_{i\leftarrow j}$), which reflects the benefit of involving multi-view cues in the encoder network. On top of the image inputs, the two networks that involve additional depth information ($Z_{i\leftarrow j}$, $Z^*_{i\leftarrow j}$) and the viewing directions (L_i, $L_{i\leftarrow j}$) both obtain better performance than the image-only versions, which leverage visibility cues and photometric cues from the inputs respectively. Our full model is able to leverage both cues from multi-view inputs and achieves the best performance.

Per-view warping. Due to potential inaccuracies in the geometry, the pixel colors of a vertex from different views may not be consistent. Directly minimizing the difference between the rendered color and the pixel color of each view will lead to ghosting artifacts, as shown in Fig. B.2. To solve this problem, we propose to apply a non-rigid warping to each view. From Fig. B.2 we can see that non-rigid warping can effectively tackle the misalignments and leads to sharper edges.

Per-vertex refinement. As shown in Fig. B.3, the image rendered using estimated SVBRDF without per-vertex refinement loses high-frequency details such as the tiny spots on the pumpkin, due to the existence of the bottleneck in our SVBRDF network. In contrast, the proposed per-vertex refinement can successfully recover these details and reproduces more faithful appearance of the object.

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, pages 40–49, 2018.

[2] Ehsan Aganj, Pascal Monasse, and Renaud Keriven. Multi-view texturing of imprecise mesh. In *ACCV*, pages 468–476, 2010.

[3] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building Rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[4] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Trans. Graph.*, 35(4):65:1–65:13, July 2016.

[5] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Transactions on Graphics*, 34(4):110:1–110:13, July 2015.

[6] Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. The digital emily project: photoreal facial modeling and animation. In *Acm siggraph 2009 courses*, pages 1–15. 2009.

[7] Neil Alldrin, Todd Zickler, and David Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *CVPR*, pages 1–8. IEEE, 2008.

[8] S. H. Baek, I. Choi, and M. H. Kim. Multiview image completion with space structure propagation. In *CVPR*, pages 488–496, 2016.

[9] Seung-Hwan Baek, Daniel S Jeon, Xin Tong, and Min H Kim. Simultaneous acquisition of polarimetric SVBRDF and normals. *ACM Transactions on Graphics*, 37(6):268–1, 2018.

[10] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM TOG*, 28:24:1–24:11, 2009.

[11] Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. Stylizing animation by example. *ACM TOG*, 32(4):119:1–119:12, 2013.

[12] P. Bérard, D. Bradley, M. Gross, and T. Beeler. Practical person-specific eye rigging. *Computer Graphics Forum*, 38, 2019.

[13] Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE TVCG*, 7(4):318–332, 2001.

[14] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106–1, 2017.

[15] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep hybrid real and synthetic training for intrinsic decomposition. *EGSR*, 2018.

[16] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructionsfrom multi-view photometric images. *ECCV*, 2020.

[17] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432. ACM, 2001.

[18] Brent Burley. Physically-based shading at disney. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pages 10:1–10:7, 2012.

[19] Ida Winifred Busbridge. *The mathematics of radiative transfer*. Number 50. University Press, 1960.

[20] Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. Real-time high-fidelity facial performance capture. *ACM Transactions on Graphics (ToG)*, 34(4):1–9, 2015.

[21] Guanying Chen, Kai Han, Boxin Shi, Yasuyuki Matsushita, and Kwan-Yee K Wong. Self-calibrating deep photometric stereo networks. In *ECCV*, 2018.

[22] Guanying Chen, Kai Han, and Kwan-Yee K Wong. Ps-fcn: A flexible learning framework for photometric stereo. In *ECCV*, 2018.

[23] Zhang Chen, Anpei Chen, Guli Zhang, Chengyuan Wang, Yu Ji, Kiriakos N. Kutulakos, and Jingyi Yu. A neural rendering framework for free-viewpoint relighting. In *CVPR*, June 2020.

[24] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *arXiv preprint arXiv:1812.02822*, 2018.

[25] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546, 2005.

[26] Robert T Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363. IEEE, 1996.

[27] M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello, and R. Scopigno. Fully automatic registration of image sets on approximate geometry. *IJCV*, 102(1-3):91–111, 2013.

[28] Massimiliano Corsini, Matteo Dellepiane, Federico Ponchio, and Roberto Scopigno. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. *CGF*, 28(7):1755–1764, 2009.

[29] Frederique Crete-Roffet, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: Perception and estimation with a new no-reference perceptual blur metric. In *HVIE*, 2007.

[30] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH*, pages 145–156. ACM Press/Addison-Wesley Publishing Co., 2000.

[31] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[32] M. Dellepiane, R. Marroquim, M. Callieri, P. Cignoni, and R. Scopigno. Flow-based local optimization for image-to-geometry projection. *IEEE TVCG*, 18(3):463–474, 2012.

[33] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image SVBRDF capture with a rendering-aware deep network. *ACM Transactions on Graphics*, 37(4):128, 2018.

[34] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Flexible svbrdf capture with a multi-image deep network. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 38(4), July 2019.

[35] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics*, 33(6):193, 2014.

[36] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *CGF*, 27(2):409–418, 2008.

[37] Sing Choong Foo. *A gonioreflectometer for measuring the bidirectional reflectance of material for use in illumination computation*. PhD thesis, Citeseer, 1997.

[38] Thomas Franken, Matteo Dellepiane, Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Minimizing user intervention in registering 2D images to 3D models. *The Visual Computer*, 21(8):619–628, 2005.

[39] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[40] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

[41] Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. Seamless montage for texturing models. *CGF*, 29(2):479–486, 2010.

[42] 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 Transactions on Graphics*, 38(4):134, 2019.

[43] Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.*, 32(6):158–1, 2013.

[44] Abhijeet Ghosh, Tim Hawkins, Pieter Peers, Sune Frederiksen, and Paul Debevec. Practical modeling and acquisition of layered facial reflectance. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–10. 2008.

[45] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2009.

[46] Paulo Gotardo, Jérémy Riviere, Derek Bradley, Abhijeet Ghosh, and Thabo Beeler. Practical dynamic facial appearance modeling and acquisition. *ACM Trans. Graph.*, 37(6), December 2018.

[47] Paulo F. U. Gotardo, Tomas Simon, Yaser Sheikh, and Iain Matthews. Photogeometric scene flow for high-detail dynamic 3d reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[48] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, pages 216–224, 2018.

[49] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, Danhang Tang, Anastasia Tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Paul Debevec, and Shahram Izadi. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Trans. Graph.*, 38(6), November 2019.

[50] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *NIPS*, 2016.

[51] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM TOG*, 35(6):231:1–231:11, 2016.

[52] C. Hernández, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *ICCV*, October 2007.

[53] Michael Holroyd, Jason Lawrence, and Todd Zickler. A coaxial optical scanner for synchronous acquisition of 3D geometry and surface reflectance. *ACM Transactions on Graphics*, 29(4):99, 2010.

[54] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, pages 2821–2830, 2018.

[55] Zhuo Hui, Kalyan Sunkavalli, Joon-Young Lee, Sunil Hadap, Jian Wang, and Aswin C Sankaranarayanan. Reflectance capture using univariate sampling of brdfs. In *ICCV*, pages 5362–5370, 2017.

[56] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: end-to-end deep plane sweep stereo. *ICLR*, 2019.

[57] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. *UIST*, pages 559–568, 2011.

[58] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. Lazyfluids: Appearance transfer for fluid animations. *ACM TOG*, 34(4):92:1–92:10, 2015.

[59] Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, 2001.

[60] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *ICCV*, pages 2307–2315, 2017.

[61] Nima Khademi Kalantari, Eli Shechtman, Connelly Barnes, Soheil Darabi, Dan B Goldman, and Pradeep Sen. Patch-based high dynamic range video. *ACM TOG*, 32(6), 2013.

[62] Yoshihiro Kanamori and Yuki Endo. Relighting humans: occlusion-aware inverse rendering for full-body human images. *ACM Transactions on Graphics*, 37(6):1–11, 2018.

[63] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018.

[64] Kaizhang Kang, Zimin Chen, Jiaping Wang, Kun Zhou, and Hongzhi Wu. Efficient reflectance capture using an autoencoder. *ACM Transactions on Graphics*, 37(4):127–1, 2018.

[65] Kaizhang Kang, Cihui Xie, Chengan He, Mingqi Yi, Minyi Gu, Zimin Chen, Kun Zhou, and Hongzhi Wu. Learning efficient illumination multiplexing for joint capture of reflectance and shape. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[66] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NIPS*, pages 365–376, 2017.

[67] Brian Karis and Epic Games. Real shading in unreal engine 4.

[68] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[69] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29, 2013.

[70] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[71] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Proceedings ofthe 2nd International Conference on Learning Representations*, 2013.

[72] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 3581–3589. Curran Associates, Inc., 2014.

[73] Joe Kniss, Simon Premoze, Charles Hansen, Peter Shirley, and Allen McPherson. A model for volume lighting and modeling. *IEEE transactions on visualization and computer graphics*, 9(2):150–162, 2003.

[74] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000.

[75] Lubor Ladicky, Olivier Saurer, SoHyeon Jeong, Fabio Maninchedda, and Marc Pollefeys. From point clouds to mesh using regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3893–3902, 2017.

[76] F. Langguth, K. Sunkavalli, S. Hadap, and M. Goesele. Shading-aware multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[77] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *CVPR*, pages 1–6, 2007.

[78] Hendrik P.A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. A silhouette-based algorithm for texture registration and stitching. *Graphical Models*, 63(4):245 – 262, 2001.

[79] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[80] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[81] 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):45:1–45:11, July 2017.

[82] Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. Materials for masses: SVBRDF acquisition with a single mobile phone image. In *ECCV*, pages 72–87, 2018.

[83] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018*, page 269. ACM, 2018.

[84] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *CVPR*, pages 2916–2925, 2018.

[85] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 37(4):68:1–68:13, July 2018.

[86] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):65, 2019.

[87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[88] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul E Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. *Rendering Techniques*, 2007(9):10, 2007.

[89] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[90] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[91] Abhimitra Meka, Christian Haene, Rohit Pandey, Michael Zollhöfer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, et al. Deep reflectance fields: high-quality facial reflectance field inference from color gradient illumination. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[92] 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.

[93] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.

[94] 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 Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[95] 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. *arXiv preprint arXiv:2003.08934*, 2020.

[96] Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. pagan: real-time avatars using dynamic textures. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

[97] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical SVBRDF acquisition of 3D objects with unstructured flash photography. In *SIGGRAPH Asia 2018*, page 267. ACM, 2018.

[98] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[99] Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. On optimal, minimal brdf sampling for reflectance acquisition. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015.

[100] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, pages 3504–3515, 2020.

[101] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte carlo methods for volumetric light transport simulation. In *Computer Graphics Forum*, volume 37, pages 551–576. Wiley Online Library, 2018.

[102] Eyal Ofek, Erez Shilat, Ari Rappoport, and Michael Werman. Multiresolution textures from image sequences. *IEEE Computer Graphics and Applications*, 17(2):18–29, 1997.

[103] A. Owens, C. Barnes, A. Flint, H. Singh, and W. Freeman. Camouflaging an object from many viewpoints. In *CVPR*, pages 2782–2789, 2014.

[104] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019.

[105] Despoina Paschalidou, Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *CVPR*, pages 3897–3906, 2018.

[106] Pieter Peers, Dhruv K Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. Compressive light transport sensing. *ACM Transactions on Graphics (TOG)*, 28(1):3, 2009.

[107] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[108] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH*, pages 75–84, 1998.

[109] Kari Pulli and Linda G. Shapiro. Surface reconstruction and display from range and color data. *Graphical Models*, 62(3):165 – 201, 2000.

[110] Stephan R Richter and Stefan Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. In *CVPR*, pages 1936–1944, 2018.

[111] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctnetFusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision*, pages 57–66. IEEE, 2017.

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

[113] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, October 2019.

[114] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[115] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.

[116] Gabriel Schwartz, Shih-En Wei, Te-Li Wang, Stephen Lombardi, Tomas Simon, Jason Saragih, and Yaser Sheikh. The eyes have it: an integrated eye and face model for photorealistic facial animation. *ACM Transactions on Graphics (TOG)*, 39(4):91–1, 2020.

[117] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, pages 519–528, 2006.

[118] Pradeep Sen, Nima Khademi Kalantari, Maziar Yaesoubi, Soheil Darabi, Dan B Goldman, and Eli Shechtman. Robust patch-based HDR reconstruction of dynamic scenes. *ACM TOG*, 31(6), 2012.

[119] Mike Seymour, Chris Evans, and Kim Libreri. Meet mike: epic avatars. In *ACM SIGGRAPH 2017 VR Village*, pages 1–2. 2017.

[120] Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steve Seitz. Regenerative morphing. In *CVPR*, pages 615–622, 2010.

[121] Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5541–5550, 2017.

[122] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, pages 1–8, 2008.

[123] Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *ACM TOG*, 27(5):159:1–159:10, 2008.

[124] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, pages 2437–2446, 2019.

[125] 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 Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019.

[126] Ioannis Stamos and Peter K. Allen. Geometry and texture recovery of scenes of large scale. *Computer Vision and Image Understanding*, 88(2):94–118, 2002.

[127] 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 Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[128] Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *SIGGRAPH*, 2019.

[129] Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Trans. Graph.*, 38(4), July 2019.

[130] Tiancheng Sun, Zexiang Xu, Xiuming Zhang, Sean Fanello, Christoph Rhemann, Paul Debevec, Yun-Ta Tsai, Jonathan T Barron, and Ravi Ramamoorthi. Light stage super-resolution: continuous high-frequency relighting. *ACM Transactions on Graphics (TOG)*, 39(6):1–12, 2020.

[131] T. Thonat, E. Shechtman, S. Paris, and G. Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *IEEE 3DV*, pages 351–359, 2016.

[132] Borom Tunwattanapong, Graham Fyffe, Paul Graham, Jay Busch, Xueming Yu, Abhijeet Ghosh, and Paul Debevec. Acquiring reflectance and shape from continuous spherical harmonic illumination. *ACM Transactions on graphics*, 32(4):109, 2013.

[133] Yochay Tzur and Ayellet Tal. Flexistickers: Photogrammetric texture mapping using casual images. *ACM TOG*, 28(3):45:1–45:10, 2009.

[134] Luiz Velho and Jonas Sossai Jr. Projective texture atlas construction for 3D photography. *The Visual Computer*, 23(9):621–629, 2007.

[135] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3D reconstructions. In *ECCV*, pages 836–850. Springer, 2014.

[136] Jinglu Wang, Bo Sun, and Yan Lu. Mvpnet: Multi-view point regression networks for 3d object reconstruction from a single image. *arXiv preprint arXiv:1811.09410*, 2018.

[137] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single rgb images. In *ECCV*, pages 52–67, 2018.

[138] Shih-En Wei, Jason Saragih, Tomas Simon, Adam W. Harley, Stephen Lombardi, Michal Perdoch, Alexander Hypes, Dawei Wang, Hernan Badino, and Yaser Sheikh. Vr facial animation via multiview image translation. *ACM Trans. Graph.*, 38(4), July 2019.

[139] Andreas Wenger, Andrew Gardner, Chris Tchou, Jonas Unger, Tim Hawkins, and Paul Debevec. Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Transactions on Graphics (TOG)*, 24(3):756–764, 2005.

[140] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE PAMI*, 29(3):463–476, 2007.

[141] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, et al. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics (TOG)*, 25(3):1013–1024, 2006.

[142] Lance Williams. Casting curved shadows on curved surfaces. In *SIGGRAPH*, volume 12, pages 270–274. ACM, 1978.

[143] Cyrus A. Wilson, Abhijeet Ghosh, Pieter Peers, Jen-Yuan Chiang, Jay Busch, and Paul Debevec. Temporal Upsampling of Performance Geometry using Photometric Alignment. *ACM Transactions on Graphics*, 29(2), March 2010.

[144] Craig M Wittenbrink, Thomas Malzbender, and Michael E Goss. Opacity-weighted color interpolation, for volume sampling. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 135–142, 1998.

[145] C. Wu, D. Bradley, P. Garrido, M. Zollhöfer, C. Theobalt, M. Gross, and T. Beeler. Model-Based Teeth Reconstruction. *ACM Transactions on Graphics (TOG)*, 35(6), 2016.

[146] Chenglei Wu, Takaaki Shiratori, and Yaser Sheikh. Deep incremental learning for efficient high-fidelity face tracking. *ACM Trans. Graph.*, 37(6), December 2018.

[147] 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, 2015.

[148] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *CVPR*, pages 1838–1847, 2018.

[149] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *NIPS*, pages 540–550, 2017.

[150] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[151] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[152] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Transactions on Graphics*, 35(6):187, 2016.

[153] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics*, 38(4):76, 2019.

[154] Zexiang Xu, Jannik Boll Nielsen, Jiyang Yu, Henrik Wann Jensen, and Ravi Ramamoorthi. Minimal brdf sampling for two-shot near-field reflectance acquisition. *ACM Transactions on Graphics*, 35(6):188, 2016.

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

[156] Shugo Yamaguchi, Shunsuke Saito, Koki Nagano, Yajie Zhao, Weikai Chen, Kyle Olszewski, Shigeo Morishima, and Hao Li. High-fidelity facial reflectance and geometry inference from an unconstrained image. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[157] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, pages 767–783, 2018.

[158] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A differential theory of radiative transfer. *ACM Trans. Graph.*, 38(6), 2019.

[159] Xiuming Zhang, Sean Fanello, Yun-Ta Tsai, Tiancheng Sun, Tianfan Xue, Rohit Pandey, Sergio Orts-Escolano, Philip Davidson, Christoph Rhemann, Paul Debevec, Jonathan T. Barron, Ravi Ramamoorthi, and William T. Freeman. Neural light transport for relighting and view synthesis. *ACM Transactions on Graphics (TOG)*, 2020.

[160] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W Jacobs. Deep single-image portrait relighting. In *CVPR*, pages 7194–7202, 2019.

[161] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics*, 33(4):155, 2014.

[162] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[163] Zhiming Zhou, Guojun Chen, Yue Dong, David Wipf, Yong Yu, John Snyder, and Xin Tong. Sparse-as-possible SVBRDF acquisition. *ACM Transactions on Graphics*, 35(6):189, 2016.