UNIVERSITY OF CALIFORNIA SAN DIEGO

# Photorealistic Digital Content Creation for Extended Reality from Sparse In-the-wild Images

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

in

Computer Science

by

Yu-Ying Yeh

Committee in charge:

Professor Manmohan Chandraker, Chair
Professor Ravi Ramamoorthi
Professor Hao Su
Professor Xiaolong Wang

2024

The Dissertation of Yu-Ying Yeh is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my Ph.D. advisor, Manmohan Chandraker, for his invaluable support throughout my entire Ph.D. journey. His unwavering support has afforded me to work on cutting-edge research with numerous exciting applications for extended reality. He has provided not only insightful guidance on my projects but also served as a life mentor. His warm words have encouraged me to become more confident in myself. Under his guidance, I was honored to receive multiple Ph.D. fellowships and internship opportunities. His mentorship has been crucial and will greatly benefit my future career and life.

I would also like to extend my thanks to Ravi Ramamoorthi, Hao Su, and Xiaolong Wang for serving on my thesis committee. Their feedback has been extremely helpful in the completion of this thesis.

During my Ph.D. journey, I had the privilege of collaborating with Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, and Zexiang Xu from Adobe Research. The project we worked on laid a crucial foundation for this thesis, specifically in creating photorealistic content (PhotoScene, detailed in Chapter 3). In addition, I had the honor of working with Ming-Yu Liu, Ting-Chun Wang, Koki Nagano, Sameh Khamis, and Jan Kautz from NVIDIA Research on a portrait relighting project (Lumos, detailed in Chapter 6). I gained significant insights into the future of content generation while working with them. Furthermore, I was fortunate to work on innovative generative frameworks for image-guided texturing (TextureDreamer, detailed in Chapter 4) with Zhao Dong, Zhengqin Li, Jai-Bin Huang, Changil Kim, Lei Xiao, Thu Nguyen-Phuoc, Numair Khan, Cheng Zhang, and Carl Marshall from Meta Reality Labs. I am grateful for the opportunity to contribute to impactful projects in 3D generative AI.

I want to extend my special thank to Zhengqin Li for his unwavering support throughout my Ph.D. studies at UC San Diego. He introduced me to the world of inverse rendering and played a significant role in several projects within this thesis. His support was pivotal in achieving numerous milestones. Additionally, I would like to thank my peers and friends–Tarun Kalluri, Rui Zhu, Ishit Mehta, Kunal Gupta, Meng Song, Sarah Wang, Ting-Wei Yu, Kai-En Lin, Alexandr

Kuznetsov, Sai Bi, Tiancheng Sun, Shilin Zhu, Lifan Wu, Zexiang Xu, Alex Trevithick, Bing Xu, Shiyang Jia, Nithin Raghavan, Xuanda Yang, Ya-Chien Chang, Tz-Ying Wu, Chih-Hui Ho, Chi-Hsin Huang, Hsuan Chang, and Yueh-Hua Wu–for their support and help. I am also grateful to Google for providing a two-year Ph.D. fellowship that supported my studies.

Finally, I would like to thank my family and my partner, Yen-Cheng Liu, for their steadfast support throughout my Ph.D. journey. Yen-Cheng's encouragement always kept me optimistic and helped me overcome challenges. I cherish the unforgettable moments we shared while reaching milestones toward completing our Ph.D. degrees.

Chapter 3 is based on the material as it appears in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022 ("PhotoScene: Photorealistic Material and Lighting Transfer for Indoor Scenes", Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, Manmohan Chandraker). The dissertation author was the primary investigator and author of this paper.

Chapter 4 is based on the material as it appears in arXiv preprint arXiv:2401.09416. ("TextureDreamer: Image-guided Texture Synthesis through Geometry-aware Diffusion", Yu-Ying Yeh, Jia-Bin Huang, Changil Kim, Lei Xiao, Thu Nguyen-Phuoc, Numair Khan, Cheng Zhang, Manmohan Chandraker, Carl S Marshall, Zhao Dong, Zhengqin Li). The dissertation author was the primary investigator and author of this paper.

Chapter 5 is based on the material as it appears in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020 ("Through the Looking Glass: Neural 3D Reconstruction of Transparent Shapes", Zhengqin Li*, Yu-Ying Yeh*, Manmohan Chandraker). The dissertation author was one of the primary investigators and authors of this paper. Zhengqin Li is a co-first author who contributed equally to the paper.

Chapter 6 is based on the material as it appears in ACM Transactions on Graphics, 2022 ("Learning to Relight Portrait Images via a Virtual Light Stage and Synthetic-to-Real Adaptation", Yu-Ying Yeh, Koki Nagano, Sameh Khamis, Jan Kautz, Ming-Yu Liu, Ting-Chun Wang ). The dissertation author was the primary investigator and author of this paper.

# VITA

| | |
|---|---|
| 2015 | Bachelor of Science and Bachelor of Arts, National Taiwan University, Taiwan |
| 2016-2017 | Research Assistant, Academia Sinica, Taiwan |
| 2017-2018 | Research Assistant, National Taiwan University, Taiwan |
| 2024 | Doctor of Philosophy, University of California San Diego, U.S.A. |

ABSTRACT OF THE DISSERTATION

Photorealistic Digital Content Creation for Extended Reality from Sparse In-the-wild Images

by

Yu-Ying Yeh

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Manmohan Chandraker, Chair

Extended Reality (XR) encompasses immersive technologies such as virtual reality (VR), augmented reality (AR), and mixed reality (MR), which blend the physical and digital worlds. For XR experiences to captivate users and seamlessly integrate with reality, photorealistic content is essential. Photorealism ensures that virtual elements convincingly interact with real-world environments, enhancing immersion and fostering a sense of presence for users. This dissertation explores various methods to facilitate the convenient and efficient creation of photorealistic digital content from images for diverse subjects.

Creating photorealistic content from images involves estimating intrinsic components from scenes, a highly challenging and ill-posed problem. To ensure photorealism, this disserta-

tion focuses on synthesizing spatially-varying Bidirectional Reflectance Distribution Functions (BRDFs) or textures, modeling complex light transport, and leveraging large-scale real-world data. Firstly, we discuss how existing priors can be used for material and lighting transfer from images to 3D scene geometry. Secondly, we explore the utilization of diffusion models pre-trained on large-scale real-world images as priors for high-quality texture synthesis and transfer to arbitrary 3D shapes with image inputs. Additionally, specialized objects like transparent shapes or portraits are addressed through learning-based approaches with synthetic data and synthetic-to-real adaptation for complex light transport and relighting to handle specific appearances.

The key contribution of this dissertation is developing efficient methods to create high-quality photorealistic content for XR with minimal human effort. Unlike prior works that depend on elaborate capture systems or extensive image sets, this dissertation achieves photorealism using just a few images easily captured from commercial mobile devices. We demonstrate diverse, high-quality photorealistic content produced by our methods, suitable for various XR applications.

# Chapter 1

# Introduction

The recent development of mobile phones, smart glasses, and spatial computing headsets enables various applications in augmented reality (AR), virtual reality (VR), and mixed reality (MR) – The era of extended reality (XR) is coming. The most important and time-consuming part of building XR applications is to generate photorealistic content.

In general, we consider a rendering pipeline to display content to the device. In this scenario, material and textures will highly determine the appearance of the content. We aim to synthesize high-quality and high-fidelity materials or textures for a given object or scene. We can render photorealistic content with rendering pipelines by modeling surface reflectance with physically-based rendering (PBR) materials. In this way, most of the surfaces can be modeled by microfacet models. However, there are special kinds of subjects that microfacet models can not describe. For example, both reflections and refractions are involved in transparent objects, and there are subsurface scatterings under the skins. It is non-trivial to create such material and texture to ensure the appearances are photorealistic. It usually requires laborious human efforts even for an experienced artist. Therefore, we aim to propose various methods and frameworks to synthesize photorealistic content from a few given images.

First, we consider the creation process as an inverse rendering problem – Estimating the decomposition of geometry, material, and lighting from input images of input objects or scenes. However, inverse rendering is a highly ill-posed problem, because there is more than

one possible decomposition that can lead to the same appearance. We therefore need to leverage priors to ensure the results are plausible. In particular, we take advantage of procedural graphs to describe diverse types of spatially varying materials. In this way, we simplify the unknowns from the number of pixels of textures to a few node parameters of graphs, yet ensure high-quality materials given from the nature of the graphs. By using procedural graphs as material prior, we can build a framework to alternately optimize for material and lighting given an indoor scene geometry configuration and a single or a few input images. In the end, we create an automatic or semi-automatic pipeline to generate digital assets from input images. However, the assumption of using procedural graphs can limit the representation of generated materials. Therefore, we look for other priors to improve the quality of the materials or textures.

Generative models learn to map a noise distribution to a complex image distribution, enabling the synthesis of high-quality, photorealistic images after training on large-scale image datasets. They can also serve as a prior to guide material and texture optimization, ensuring photorealistic appearances. Thus, we can create a framework that first fine-tunes the generative model on a few images of an object whose appearance interests us. Then, we distill the appearance from this fine-tuned model to the target shape, creating high-quality, photorealistic, and relightable textures using PBR materials within a differentiable rendering pipeline. In this pipeline, we typically assume a microfacet model for PBR materials. However, many subjects, such as transparent objects and portraits, are not well-described by the microfacet model. Therefore, we propose leveraging synthetic data to capture light transport for photorealistic appearances.

To create photorealistic content for transparent objects and human portraits, we must address specific reflectance and light transport properties. Transparent objects involve light reflecting and refracting on their surfaces, while human skin exhibits subsurface scattering. Optimizing implicit parameters from a sparse set of images is non-trivial, as the appearance of transparent objects depends on environmental illumination, and the surfaces of portraits consist of micro-displacements. This complexity makes physically-based rendering time-consuming and

prone to noise. For transparent objects, we leverage synthetic data to employ a learning-based approach with a physically-motivated network for shape reconstruction. For human portraits, we focus on target editing tasks, such as relighting, using synthetic datasets generated by a physically-based rendering engine. This approach allows us to bypass the need for detailed micro-geometry modeling and directly synthesize relighting results. However, real-world portrait images exhibit diverse skin colors, hairstyles, clothing, and other features. Consequently, synthetic data alone cannot ensure sufficient generalizability to real portraits. To address this, we implement synthetic-to-real domain adaptation to mitigate domain gaps and enhance the realism and applicability of our models.

## 1.1 Outline

The thesis is organized as follows. In Chapter 2, we first provide some background on the relevant topics of photorealistic content creation. In the following Chapters, we introduce different ways to generate photorealistic 3D content and handle photorealistic editing for special types of subjects.

In Chapter 3, we introduce PhotoScene, a framework designed to generate a photorealistic digital replica of a scene from the input image(s) with CAD geometry that is approximately aligned, either automatically reconstructed or manually specified. Our approach utilizes procedural material graphs to model scene materials, ensuring photorealism and resolution independence. By optimizing the parameters of these graphs, along with texture scale, rotation, and scene lighting, we aim to closely match the input image using a differentiable rendering layer.

In Chapter 4, we introduce TextureDreamer, an innovative technique for image-guided texture synthesis that facilitates the transfer of relightable textures from a limited set of input images (ranging from 3 to 5) to diverse 3D shapes spanning various categories. TextureDreamer enables the migration of intricate, highly detailed textures from real-world contexts to arbitrary objects, leveraging just a handful of casually captured images. This approach holds the

potential to greatly democratize texture creation processes. Our methodology is inspired by recent advancements in diffuse models, incorporating personalized modeling to extract texture information, score distillation for nuanced appearance synthesis, and explicit geometry guidance via ControlNet. Through integration and key modifications, we notably enhance texture quality.

In Chapter 5, we suggest a physics-driven network designed to reconstruct the 3D shape of transparent objects using several images taken with a smartphone camera, under any known environment map. Our unique contributions consist of a normal representation that allows the network to handle intricate light transport through localized computations, a rendering layer capable of simulating refractions and reflections, a specially crafted cost volume for refining the normals of transparent shapes, and a feature mapping that relies on predicted normals to reconstruct 3D point clouds. To train our model to understand refractive light transport from various perspectives, we generate a synthetic dataset.

In Chapter 6, we discuss synthesizing photorealistic relighting effects on human portraits. Traditionally, achieving high-quality results involves training deep neural networks with a dataset of precise input-output pairs obtained using a costly light stage. However, the high expense and effort restrict this method to well-equipped labs. To overcome this, we propose a new method that matches state-of-the-art relighting techniques without needing a light stage. Our strategy includes two key conditions for success: mimicking physical relighting behaviors and producing photorealistic outputs. We simulate a virtual light stage as a synthetic dataset for network training, which is created by physically-based rendering on diverse 3D synthetic humans across various environment maps. Additionally, we introduce a novel synthetic-to-real technique to enhance the network's output photorealism. This approach not only meets top standards but also improves on previous methods by allowing controllable effects like glares on glasses and providing more consistent results in relighting videos.

Finally, we summarize our contributions in Chapter 7 and discuss potential future directions inspired by the content in this thesis for photorealistic content creation.

# Chapter 2

# Background

In the traditional computer graphics pipeline, 3D meshes with photorealistic materials are used to enable photorealistic rendering. With a physically accurate rendering engine, photorealistic images can be produced under desired lighting conditions. However, creating high-quality 3D content is time-consuming and requires significant effort, even for experienced artists. Consequently, researchers have begun addressing the ill-posed problem of estimating 3D intrinsic components from images to streamline this process.

It has been a long-lasting problem that people trying to reconstruct 3D objects and scenes from photos or videos. Photometric stereo [242] recovers surface normals from images captured in unknown lighting conditions. Stereo matching [12] estimates depth from multiple images of the same object or scene. Structure-from-Motion (SfM) [132] reconstructs 3D point clouds from a set of images of a scene by estimating camera poses and triangulation from matching points. Recently, deep neural networks have been used to estimate 3D meshes of objects from single view [244, 269, 61, 222] or multi-view images [35, 247, 228]. Some works [158, 261] can even reconstruct the geometry of an entire indoor scene. While most of the methods only aim to estimate geometry from images, some works [95, 189] propose to estimate textured meshes. However, there is still a gap in achieving photorealistic appearances due to the lack of estimation of spatially-varying bidirectional reflectance distribution function (SVBRDF) for materials and complex lighting. Shape, material, and lighting can be optimized via a differentiable

rasterization [153] or Monte Carlo rendering and denoising [71] pipeline with enough number (usually over a hundred) of input images. To achieve a more practical setup – only using a sparse number of images, we propose PhotoScene [254], which transfers SVBRDF and lighting from sparse images to 3D meshes of an indoor scene with the aid of procedural material prior and part-based differentiable rendering module in Chapter 3. Another drawback of the prior works for modeling SVBRDFs is that usually the surfaces are assumed to be microfacet. Instead, we propose a solution to reconstruct shapes of transparent objects from a sparse number of images by a learning-based approach with a physically-motivated neural renderer, as detailed in Chapter 5.

Recent developments in neural networks enable more ways to represent 3D content for photorealistic rendering. Implicit 3D representations [140, 248, 268, 230], have been proposed to represent shapes via a multilayer perceptron (MLP) which encodes occupancy [140] or signed distance field [248] for each location query. Neural radiance fields [144] (NeRF) have been proposed for novel view synthesis with view-dependent rendering effects. Some variants [268] can distill volumetric geometry, SVBRDF, and lighting from a NeRF representation, while NeuS [230] learns neural implicit surfaces with volume rendering. The implicit 3D representation can be rendered by sphere tracing [70, 129] or ray marching [138]. There are also various neural rendering methods that enable photorealistic rendering with neural networks with different 3D representations such as neural 3D point clouds [39, 249]. Recently, 3D Gaussian splatting [104] has shown to be a more efficient way to represent the radiance field which enables real-time rendering with high photorealism. The implicit 3D representations or 3D Gaussian splatting can be used to perform photorealistic rendering for scenes reconstructed from a large number (usually over a hundred) of images or can be integrated into an optimization or learnable framework. Our focus is instead more on synthesizing photorealistic content from sparse image sets. We demonstrate that the implicit representation can be used to represent relightable textures for an arbitrary 3D mesh in an optimization pipeline in Chapter 4.

To solve an ill-posed problem such as inverse rendering or build up an editing (*e.g.* relighting) system for photorealistic rendering from sparse views, we can take advantage of

synthetic data to learn the intrinsic properties from input images. However, synthetic datasets are usually created by physically-based rendering with pre-defined 3D assets including shapes, materials, and lighting. Due to the complexity of the real-world environment, it is difficult to build realistic scenes to produce photo-like renderings. There are a few datasets [126, 276, 255] that are proposed for inverse rendering, however, we can still identify the domain gap between rendered images and real-world photos. Researchers have proposed various methods to close this domain gaps for a variety of tasks, including semantic segmentation [48, 79], 3D human pose estimation [46], animal pose estimation [117], SVBRDF estimation [226], depth estimation [8, 271], vehicle re-Id [115], and multi-view portrait view synthesis and relighting [217]. In Chapter 6, we proposed a framework for single-image or video portrait relighting by using synthetic data and synthetic-to-real domain adaptation to produce high-quality relighting effects.

On the other hand, generative models (*e.g.* generative adversarial network [62], variational autoencoders [106], or diffusion models [77]) learn a mapping from noise distribution to the real data distribution. This enables generations of highly photorealistic images from noises or particular conditions. Recently, the success of latent diffusion models [187] inspired a wide range of downstream tasks to synthesize unprecedented-quality images or 3D content. The foundation models learned from a large-scale dataset (*e.g.* LAION-5B [192], a billion-level dataset) serve as generative priors to guide the optimization to follow real-world data distribution. In Chapter 4, we demonstrate that the latent diffusion models can serve as a distillation source to optimize for high-quality textures for 3D assets given a sparse image set.

More detailed related work for Chapters 3, 4, 5 and 6 will be provided in Chapters 3.2, 4.2, 5.2 and 6.2, respectively.

# Chapter 3

# Photorealistic Material and Lighting transfer for Indoor Scenes

## 3.1 Introduction

A core need in 3D content creation is to recreate indoor scenes from photographs with a high degree of photorealism. Such photorealistic "digital twins" can be used in a variety of applications including augmented reality, photographic editing and simulations for training in synthetic yet realistic environments. In recent years, commodity RGBD sensors have become common and remarkable progress has been made in reconstructing 3D scene geometry from both single [52, 215] and multiple photographs [252], as well as in aligning 3D models to images to build CAD-like scene reconstructions [89, 10, 85, 158]. But photorealistic applications require going beyond the above geometry acquisition to capture material and lighting too — to not only recreate appearances accurately but also visualize and edit them at arbitrary resolutions, under novel views and illumination.

Prior works assign material to geometry under the simplifying assumptions of homogeneous material [89] or single objects [162]. In contrast, we deal with the challenge of ascribing spatially-varying material to an indoor scene while reasoning about its complex and global interactions with arbitrary unknown illumination. One approach to our problem would be to rely on state-of-the-art inverse rendering methods [121, 126] to reconstruct per-pixel material properties and lighting. However, these methods are limited to the viewpoint and resolution of


**Figure 3.1.** Given an input photo and a coarsely aligned 3D scene model, PhotoScene automatically infers high-quality spatially-varying procedural materials and scene illumination to closely match scene appearance. The reconstructed materials are resolution-independent (see zoom insets) and ascribed to the full 3D geometry, to create a high-quality photorealistic digital twin that can be rendered under novel views and lighting.

the input photograph, and do not assign materials to regions that are not visible (either outside the field of view or occluded by other objects). Instead, we posit that learned scene priors from inverse rendering are a good initialization, whereafter a judicious combination of expressive material priors and physically-based differentiable rendering can solve the extremely ill-posed optimization of spatially-varying material and lighting.

In this chapter, we use procedural node graphs as compact yet expressive priors for scene material properties. Such graphs are heavily used in the content and design industry to represent *high-quality, resolution-independent* materials with a compact set of optimizable parameters [1, 4]. This offers a significant advantage: if the parameters of a procedural graph can be estimated from just the observed parts of the scene in an image, we can use the full graph to ascribe materials to the entire scene. Prior work of Shi et al. [200] estimates procedural materials, but is restricted to fully observed flat material samples imaged under known flash

illumination. In contrast, we demonstrate that such procedural materials can be estimated from partial observations of indoor scenes under arbitrary, unknown illumination.

We assume as input a coarse 3D model of the scene with possibly imperfect alignment to the image, obtained through 3D reconstruction methods [10, 85, 158], or manually assembled by an artist. We segment the image into distinct material regions, identify an appropriate procedural graph (from a library) for each region, then use the 3D scene geometries and their corresponding texture UV parameterizations to "unwarp" these pixels into (usually incomplete) 2D textures. This establishes a fully differentiable pipeline from the parameters of the procedural material via a physically-based rendering layer to an image of the scene, allowing us to backpropagate the rendering error to optimize the material parameters. In addition, we also estimate rotation and scale of the UV parameterization and optimize the parameters of the globally-consistent scene illumination to best match the input photograph.

As shown in Fig. 3.1 our method can infer spatially-varying materials and lighting even from a single image. Transferring these materials to the input geometry produces a fully relightable 3D scene that can then be rendered under novel viewpoint or lighting. Since procedural materials are resolution-invariant and tileable, we can render closeup views that reveal fine material details, without having observed these in the input photograph. This goes significantly beyond the capabilities of current scene-level inverse rendering methods and allows for the creation of high-quality, photorealistic replicas of complex indoor scenes.

## 3.2 Related Works

### 3.2.1 Material acquisition and recognition

High-quality materials have been estimated in many prior works, using both single [5, 43, 122, 75] or multiple [44, 55] input images. Most of the above methods estimate materials for planar samples, as opposed our inputs that are unconstrained images of complex indoor scenes. While material recognition methods have been proposed to classify image regions into

material categories [16], they do not yield parametric materials that could be used for relighting and view synthesis. In recent years, several methods have been proposed to use procedural graphs as materials priors and estimate their parameters to match the appearance of captured images [66, 83, 200]. In particular, we use MATch[200], a differentiable procedural material model based on Substance node graphs, to constrain our materials to the SVBRDF manifold. However, the above methods only consider flat material samples captured under known flash illumination, while our goal is significantly more challenging – our inputs are photos of indoor scenes with complex geometry, lit by unknown spatially-varying illumination, with scene layout and occlusions leading to incomplete textures with arbitrary scales and rotation.

### 3.2.2 Inverse rendering of indoor scenes

Our problem may be seen as an instance of inverse rendering [136, 165], but we must estimate materials that are amenable for rendering under novel views and lighting, as well as editability. Several approaches have been proposed for inverse rendering for objects [128, 197, 206, 221, 123, 190], but our focus is on indoor scenes, which have been considered in both early [13, 99, 100] and recent [121, 196] works. A convolutional neural network with a differentiable rendering layer estimates depths, per-pixel SVBRDF and spatially-varying lighting in [121] using a single input image. We use their material and lighting outputs as our initialization and to aid our differentiable rendering losses in image space, but go further to assign procedural materials that can be used for rendering novel views and estimate lighting that is consistent across views. Recent work has applied differentiable rendering [119] to recover spatially-varying reflectance and lighting given photos and 3D geometry [11, 159]. However, these methods estimate per-vertex BRDFs and require high-quality geometry as input, while our procedural models regularize scene materials, allowing us to infer them from only coarsely aligned 3D models and to re-render novel viewpoints with material detail that was not observed in the input photos.

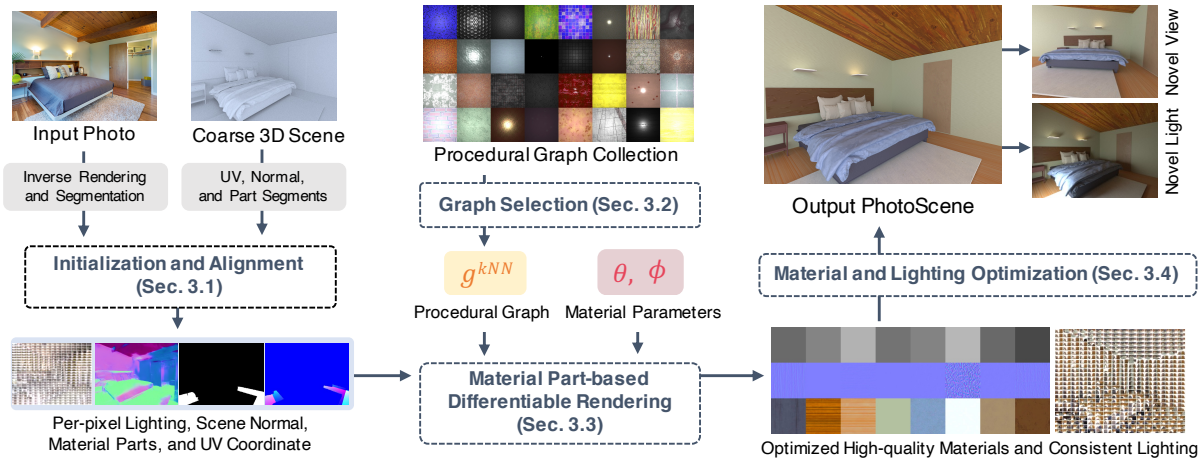



**Figure 3.2.** The PhotoScene framework. From the input photo(s) and 3D scene model, we estimate scene normals and lighting via an inverse rendering method, and compute material parts and align them to the model UVs and part segments. We model scene materials with procedural graphs. For each material part, we identify an appropriate graph from a collection and use a differentiable rendering module to optimize for the graph and UV transformation parameters. We also refine the initial lighting. Assigning the optimized materials and lighting to the input 3D model gives us our output PhotoScene—a renderable 3D scene that matches the appearance of the input photos.

### 3.2.3 Material transfer from photographs

LIME [139] clones the homogeneous material from a single color image. Material Memex [90] and Unsupervised Texture Transfer [232] exploit correlations between part geometries and materials, or across patches for material transfer to objects. PhotoShape [162] assigns photorealistic materials to 3D objects through material classifiers trained on a synthetic dataset. In contrast, we seek material transfer in indoor scenes, which is significantly harder since image appearances entwine material properties with complex light transport. The material suggestion system of [29] textures synthetic indoor scenes with high-quality materials, but based on a set of pre-defined rules for local material and global aesthetics, whereas we must solve challenging inverse problems in a differentiable framework to match the appearances of real images.

### 3.2.4 Indoor scene 3D reconstruction

Many works reconstruct indoor 3D scene geometry (objects and room layout) from single images [89, 85, 158] or RGBD scans [10], but either do not address material and lighting

estimation, or use heuristics like median color to assign diffuse textures [89]. Our work focuses on reconstructing high-quality material and lighting from input photos and is complementary to geometric reconstruction methods; indeed, we can leverage these methods to build our input coarse 3D model. Like us, Plan2Scene [227] also aims to reconstruct textured 3D scenes, but is limited to diffuse textures and constrained by the quality of the texture synthesis model. In contrast, by optimizing a procedural material model, we are able to recover high-quality non-Lambertian materials; we also optimize for illumination and hence better match the input image appearance.

## 3.3 Proposed Method

Our method starts from an input image (or multiple images) of an indoor scene and a roughly matching scene reconstruction (automatic or manual). Our goal is to obtain high-resolution tileable material textures for each object, as well as a globally consistent lighting for the scene.

The method consists of four high-level stages, as shown in Fig. 3.2. We compute an initial estimate of scene normals and lighting. We also find material parts, and align them between the input and rendered image, so that each material part can be optimized separately. Next, we choose a *material prior* for each material part, in the form of a procedural node graph that produces the material's textures (albedo, normal and roughness) given a small set of parameters. Finally, we optimize the parameters of all materials as well as the lighting in the scene. Below we describe this in detail.

### 3.3.1 Initialization and Alignment

In the initialization step, we obtain estimates of normals and lighting from the input image(s) that guide the subsequent optimization. Next, since the synthetic scene is composed of elements that are not perfectly aligned with the input photograph(s), we warp the pixels rendered from the geometry to best fit the scene structure in the input photograph *per material part*. If

there are multiple input images per scene available, we perform consensus-aware view selection (see Sec. 3.3.5 for details).

**Pixel-level normals and lighting initialization**

We use the pretrained inverse rendering network (InvRenderNet) from Li et al. [121] to obtain spatially-varying incoming lighting estimates $\mathscr{L}^{\text{inv}}$ and per-pixel normals $\mathscr{N}^{\text{inv}}$ to guide the material optimization in pixel space (Sec. 3.3.3). We do not use the estimated albedo and roughness from InvRenderNet, except as baselines for comparison, as shown in Fig. 3.13.

**Material part mask and mapping**

For each material part, our method requires a mask $M_{\text{photo}}$ to indicate the region of interest in the input image, and another mask $M_{\text{geo}}$ to indicate the same in the synthetic image. The latter mask is trivially available by rendering the synthetic geometry. To obtain $M_{\text{photo}}$ automatically, we make use of predictions from MaskFormer [34] as proposals, and find the mapping by computing maximum intersection over union (IoU) with respect to $M_{\text{geo}}$ of all material parts. Semantic and instance labels (when available) can be used to reduce the proposals. To obtain more robust results, manually segmented masks can be taken as $M_{\text{photo}}$ instead. More details can be found in Sec. 3.3.5.

**Geometry/photo alignment and warping**

To handle misalignment between material parts in the input image and the geometry, we compute an affine pixel warp. We obtain pixel locations $\mathbf{x}_s^*$ to sample values from $M_{\text{geo}}$ of the geometry to warp to the material mask $M_{\text{photo}}$ at pixel $\mathbf{x}_t$ as

$$\mathbf{x}_s^* = \frac{\mathbf{x}_t - \mathbf{c}_p}{\mathbf{l}_p} \cdot \mathbf{l}_g + \mathbf{c}_g \ , \tag{3.1}$$

where $\{\mathbf{c}_g, \mathbf{l}_g\}$ and $\{\mathbf{c}_p, \mathbf{l}_p\}$ are the centers and sizes of bounding boxes around the masks. As the affine warp is imperfect, some pixels will not have a correspondence and will be dropped. Please

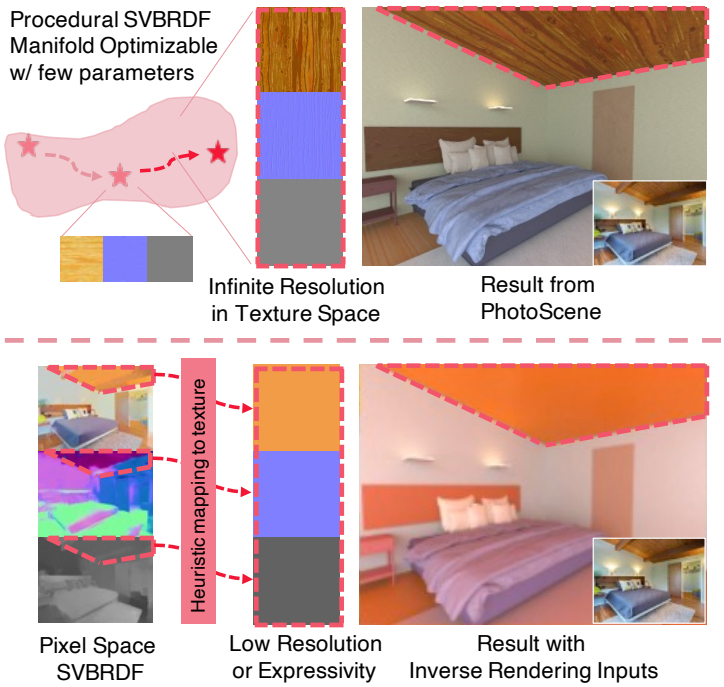



**Figure 3.3.** An expressive material prior optimizable with a few parameters allows recreating photorealistic appearances, in contrast to heuristics that may rely solely on pixel-space inverse rendering.

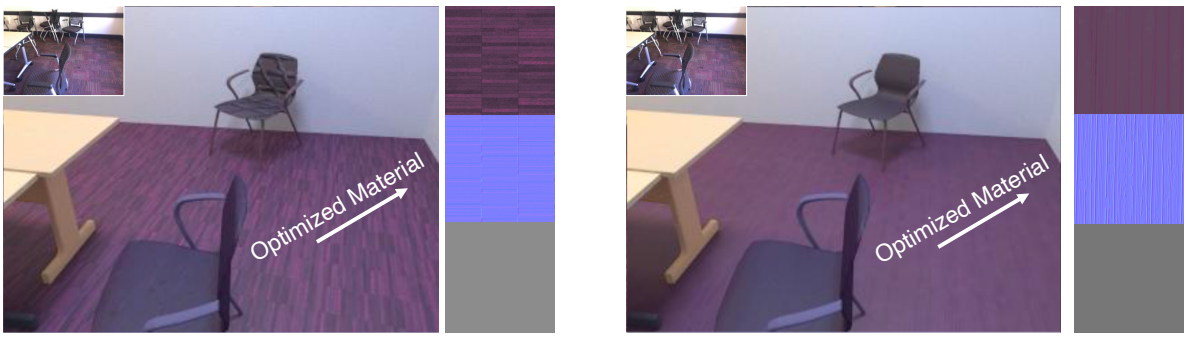
see Sec. 3.3.5 for details.

At the end of this initialization and alignment step, we obtain per-material part UVs and corresponding masks in the input and synthetic images, as well as an initial estimate of scene lighting and normals.

### 3.3.2 Material Prior: Procedural Node Graphs

Modeling spatially-varying materials as 2D textures is difficult due to the ill-posed nature of the problem: the textures may not be fully observed in the input image, and they are lit by uncontrolled illumination. Therefore, a key step of our method is to constrain materials to lie on a valid SVBRDF manifold, by specifying a *material prior* that is expressive, yet determined by a small number of parameters. This material prior must be differentiable to allow parameter optimization via backpropagation. This is in contrast to a bottom-up approach that might rely on pixel space outputs from inverse rendering (see Fig. 3.3 and 3.13).

We use MATch [200], a material prior based on differentiable procedural node graphs.




kNN-based Procedural Graph Selection  Material Classifier-based Procedural Graph Selection

**Figure 3.4.** Graph selection with kNN versus material classifier.

Their implementation provides 88 differentiable procedural graphs that model high quality spatially-varying materials, each with a unique set of parameters. For our purposes, these graphs are simply differentiable functions from a parameter vector $\theta$ to albedo, normal and roughness textures $A_{uv}$, $N_{uv}$, $R_{uv}$. We add an additional offset parameter for the albedo output from the graphs to more easily control the dominant albedo colors. We select 71 graphs that are representative of indoor scenes as our graph collection $\{g^1, ..., g^{71}\}$. We augment this set with a homogeneous material, used for untextured parts.

**Material graph selection**

For each material part in the image, we need to choose an appropriate procedural node graph from the library. We address this by nearest neighbor search using a VGG feature distance. Specifically, we sample 10 materials from each of the 71 graphs with random parameters, resulting in 710 exemplar material maps $\{(A|N|R)^1_{uv}, \cdots, (A|N|R)^{710}_{uv}\}$. We render the part using each exemplar with our differentiable renderer (Sec. 3.3.3), resulting in 710 render-graph pairs $\{(\tilde{\mathscr{I}}^1_{rend}, g^1), (\tilde{\mathscr{I}}^2_{rend}, g^1), ...(\tilde{\mathscr{I}}^{710}_{rend}, g^{71})\}$, then select the $k$ $(= 21)$ most similar renderings to the input using a masked VGG distance (Eq. 3.9), which vote for their corresponding graphs and we pick the graph $g^{kNN}$ with the most votes.

We also experiment with predicting material super classes (e.g. wood, plastic, etc.) using a pretrained classifier and then selecting a graph from the class, but find the kNN search less

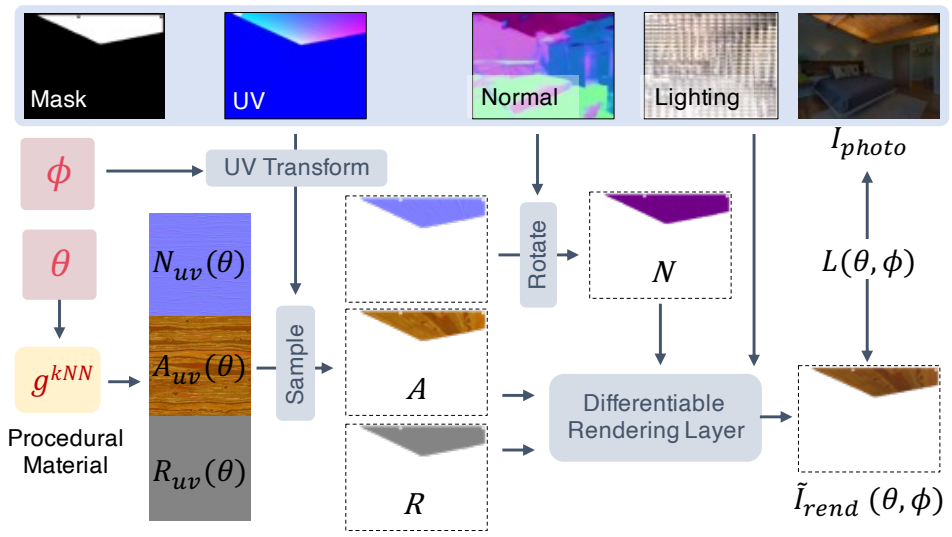



**Figure 3.5.** Given a material part mask, UV, scene normals and lighting (top), we construct a fully differentiable pipeline from material graph ($\theta$) and UV transformation ($\phi$) parameters via a texture-to-image mapping and differentiable rendering layer to a rendered image. We optimize for these parameters by comparing this rendering to the input photo.

susceptible to errors (Fig. 3.4). For small parts where it is difficult to observe spatial variations, we use homogeneous materials.

### 3.3.3 Material Part Differentiable Rendering

Our framework leverages a *material part differentiable rendering module* as a way to predict the appearance of a part given its texture-space material maps $(A|N|R)_{uv}$, pixel-space geometry $\mathscr{N}^{\text{inv}}$, and local lighting $\mathscr{L}^{\text{inv}}$. The differentiable rendering module can be used for optimization through back-propagation. We use it to optimize for material parameters in Sec. 3.3.4.

During rendering, we use a spatially-varying grid of incoming light environment maps [121] as our lighting representation, which allows the operation of the rendering module to remain local; no additional rays need to be traced by the rendering process, which is crucial for efficiency.

Our differentiable rendering module is shown in Fig. 3.5. Our material model is a physically-based microfacet BRDF [96]. The rendering module takes per-pixel texture (UV)

coordinates, to sample material textures $A_{uv}$, $N_{uv}$, $R_{uv}$ generated from the material prior using parameters $\theta$ as $g^{kNN}(\theta)$. The normals need to be rotated into the local shading frame of a given point on the material part. The rendering module then uses per-pixel material parameters $A$, $N$, $R$ and spatially-varying local incoming lighting $L$ to render the image pixels $\tilde{I}_{\text{rend}}$:

$$A, R = \quad \textbf{Sample}_{UV}(A_{uv}(\theta), R_{uv}(\theta)), \tag{3.2}$$

$$N = \quad \textbf{Rot}(\textbf{Sample}_{UV}(N_{uv}(\theta))), \tag{3.3}$$

$$\tilde{\mathscr{I}}_{\text{rend}} = \quad \textbf{RenderLayer}(A, N, R, L). \tag{3.4}$$

As the originally assigned UV coordinates might not have the optimal scale and orientation to apply the corresponding material, we apply texture transformation parameters $\phi$ (rotation, scale, and translation) to map original coordinates $UV^0$ to more appropriate ones $UV$.

$$UV = \quad \textbf{UVTransform}(UV^0, \phi). \tag{3.5}$$

Fig. 3.10 provides visual examples of the importance of considering rotation and scale in our differentiable rendering.

### 3.3.4 Material and Lighting Optimization

Images conflate lighting, geometry, and materials into an intensity value. To better disambiguate lighting from material, we adopt a two-step process. First, we use our initial spatially-varying lighting prediction from InvRenderNet to optimize the materials for each object. Next, we perform a globally consistent lighting optimization to refine our illumination. Finally, we optimize the materials once more, this time using our refined lighting. This procedure reduces the signal leakage between material and lighting.

**Material optimization**

There is no exact correspondence between the rendered and reference pixels. Thus, we compute the absolute difference $\mathscr{L}_{stat}$ of the statistics (mean $\mu$ and variance $\sigma^2$) of the masked pixels of the part of interest to optimize both material prior parameters $\theta$ and UV transformation parameters $\phi$:

$$\mathscr{L}_{\text{mean}} = |\mu(\mathscr{I}_{\text{photo}} \cdot M_{\text{aln}}) - \mu(\tilde{\mathscr{I}}_{\text{rend}} \cdot M_{\text{aln}})|, \tag{3.6}$$

$$\mathscr{L}_{\text{var}} = |\sigma^2(\mathscr{I}_{\text{photo}} \cdot M_{\text{aln}}) - \sigma^2(\tilde{\mathscr{I}}_{\text{rend}} \cdot M_{\text{aln}})|, \tag{3.7}$$

$$\mathscr{L}_{\text{stat}} = \mathscr{L}_{\text{mean}} + \mathscr{L}_{\text{var}}, \tag{3.8}$$

where $M_{\text{aln}}$ is the resulting aligned mask from the alignment step.

Using this statistics loss encourages matching color distributions but not spatially-varying patterns. To further match the patterns, we add a masked version of VGG loss $L_{vgg}$ [208]. Let $\tilde{C}_l$ and $C_l$ be the normalized VGG feature maps of $\tilde{\mathscr{I}}_{\text{rend}}$ and $I_{\text{photo}}$ extracted from layer $l$[1], we apply mask $M_{aln}$ on the sum of upsampled L2 difference of normalized feature maps $\tilde{C}_l$ and $C_l$ and compute the mask-weighted average among the pixels $x$:

$$\mathscr{L}_{\text{vgg}} = \frac{1}{\sum_x M_{\text{aln}}} \sum_x M_{\text{aln}} \left( \sum_l \text{Up}\left(\tilde{\mathscr{C}}_l - \mathscr{C}_l\right)^2 \right). \tag{3.9}$$

We also try a masked style loss based on the Gram matrices of VGG features [59], but find that it does not provide significant improvement over $L_{\text{stat}}$ and $L_{vgg}$. Therefore, we use the following loss for material optimization:

$$\mathscr{L}_{\text{total}} = \alpha \mathscr{L}_{\text{stat}} + \beta \mathscr{L}_{\text{vgg}}. \tag{3.10}$$

Rather than jointly optimizing for material and UV parameters, we find that convergence is more

[1]The layers used here are relu1_2, relu2_2, relu3_3, relu4_3, relu5_3.


**Figure 3.6.** Example of lighting optimization result using $N = 2$ ceiling lights and one environment map lighting. The optimized lighting is close to original ScanNet images.

stable with alternately searching for UV parameters in a discretized space and optimizing for material graph parameters. Spatially-varying roughness parameters are difficult to optimize in a single view due to limited observations of highlights, so we replace the roughness output from the graph with a single mean value during optimization (the final result can still use the full roughness textures).

### Globally consistent lighting optimization

To estimate globally consistent lighting, we represent indoor lighting as $N$ area lights and one environment light which may be observed through the windows. We optimize for RGB intensities for each light source. For scenes without light source annotations, we uniformly place area lights on the ceiling every 3 meters of distance. With the materials we previously optimized, we render images with each single light source turned on. We compute the RGB intensities by comparing these renderings with the input view using least squares. Specifically, with $N$

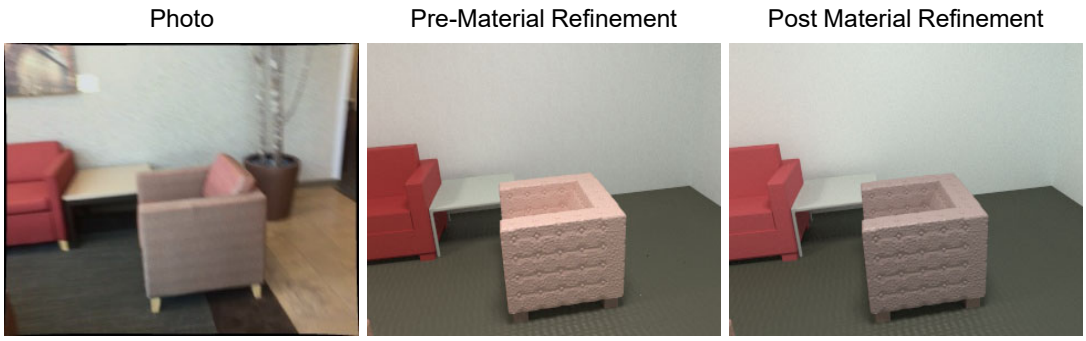



**Figure 3.7.** Second-round material refinement successfully corrects the inaccurate reflectance values estimated in the first round.

area lights (including ceiling lights and lamps) in a room with $V$ input images, we solve for $3 \times (N+1)$ RGB coefficients $x_r, x_g, x_b \in \mathbb{R}^{N+1}$ (the $+1$ refers to an environment light visible through windows). We use these coefficients to re-weight the intensities of each light source. Fig. 3.6 demonstrates examples of rendering under selected views with each light source and the final combined optimized lighting. We additionally optimize for relative exposure values under different views, since they may vary over a video acquired using commodity cameras.

**Material reoptimization**

With the refined globally-consistent light sources, we re-optimize the materials to improve our results. We render a new spatially-varying incoming lighting grid $L^{\text{global}}$ from the synthetic scene with the optimized light sources and use the same optimization loss as Sec. 3.3.3. We only optimize for a homogeneous re-scaling of the albedo and roughness maps in this round, as the spatially-varying patterns are already correctly optimized by the first iteration. Fig. 3.7 demonstrates that this refinement step can rectify inaccurate material parameters caused by albedo-lighting ambiguities in the inverse rendering network.

### 3.3.5 More Details on Initialization and Alignment

**Consensus-aware view selection**

When a video sequence is available as input, we subsample views that are at least $30^\circ$ or $1m$ apart, then choose the optimal view among them for optimizing each material part. We

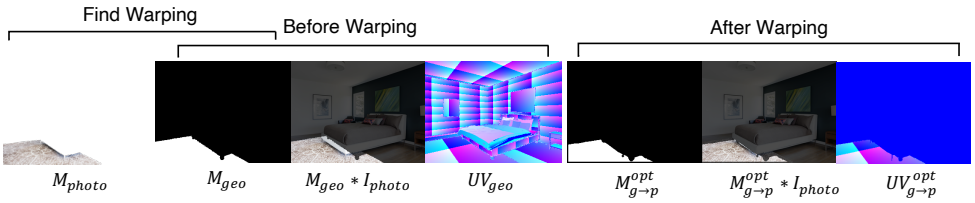



**Figure 3.8.** Example of part segmentation matching and UV warping between geometry and input image.

choose the best view based on three criteria – coverage, field-of-view and consensus. We expect good material transfer from those input images where a substantial number of pixels from the material part are observed. To ensure they occupy a favorable field-of-view, we weigh the number of pixels with a Gaussian, $G$, centered at the middle of the image and with variance one-fourth of the image dimensions. Finally, the goodness of a material part in a given view is also determined by the number of other views, $n_i$, where material estimates are in consensus, which is determined as the L2-norm of the mean and standard deviations of the per-pixel albedo and roughness predictions from InvRenderNet. We choose the view with the highest value of $n_i \cdot \sum(G \odot M_{photo})$ as the one to use for material transfer.

**Material part mask and mapping**

We regard material part segmentation as non-trivial, since material parts are ambiguous, e.g. table legs can be treated as separated parts or same part as the entire table. We found that the instance-based segmentation from MaskFormer already provides robust candidates which can later be refined by the mapping and alignment with geometry mask $M_{geo}$. Again, we can always provide better segmentation from manually labeling or existing dataset.

When MaskFormer does not detect a valid mask or 3D shapes have too small parts or highly different geometries from the image, we cannot find a large enough mask. We determine these failure situations by setting a threshold on the number of valid pixels inside a mask which can be used for optimization, and simply compute median values on the valid pixels, or on geometry mask $M_{geo}$ if no valid pixels at all. To be specific, we first compute a per-pixel weight

map $W_{aln}$ by the dot product between aligned normal from InvRenderNet $N^{inv}$ and normal from geometry $N_{geo}$ and then define the valid pixels by computing the number of pixels with the above dot product larger than 0.95 as $J$ and only run our optimization if $J \geq 500$, otherwise, we compute median for small masks where $J < 500$. If there is no mask candidate being matched by IoU, we simply use $M_{geo}$ to compute median.

The weight map $W_{aln}$ is also multiplied with $M_{aln}$ to obtain a weighted mask when computing mask-based losses during optimization.

**Alignment and warping**

Let $M_{geo}$ be the 2D material part mask rendered from geometry under corresponding views and $M_{photo}$ be the mask for the reference photo. We first decompose $M_{geo}$ and $M_{photo}$ into sub-masks $\{M^i_{geo}\}$ and $\{M^j_{photo}\}$ which represents a single instance (if there are multiple instances), and search for matching instance pairs by the highest mIoU values on *soft* instance submasks. If semantic labels for both photo and geometry are available, we can use it to reduce the sub-masks by selecting corresponding semantics. Here *soft* means we apply a Gaussian filter on the instance submask with mean set as the center of mask bounding box and standard deviation set as half of width and height of bounding box, respectively.

After finding the matching pairs of part instances, we need to find the warping relationship between $M_{geo}$ and $M_{photo}$ so that we can warp $UV_{geo}$ to $\widehat{UV}_{geo} \approx UV_{photo}$, which is used to sample material parameters from $UV$ space to image space in our material part-based differentiable rendering module. We formulate the warping as scaling and translation from bounding box $B_{geo}$ of $M_{geo}$ to bounding box $B_{photo}$ of $M_{photo}$ to avoid unnecessary rotations. Let $c_g$ and $l_g$ be the center and size of $B_{geo}$ and $c_p$ and $l_p$ be the center and size of $B_{photo}$. While $c_g$, $c_p$ and $l_g$, $l_p$ can be computed by minimum and maximum pixel locations in $x$ and $y$ directions of $M_{geo}$ and $M_{photo}$, we can further find optimal $c^*_g$ and $l^*_g$ by optimizing intersection-over-union:

$$\max_{c_g, l_g} \frac{M_{photo} \cap \widehat{M}_{geo}}{M_{photo} \cup \widehat{M}_{geo}}, \tag{3.11}$$

to ensure higher percentage of overlap between $\widehat{M}_{geo}$ (the warped $M_{geo}$) and $M_{photo}$.

We warp the $UV$ map $\widehat{UV}_{geo}$ by

$$x_s^* = (x_t - c_p)/l_p * l_g^* + c_g^*, \tag{3.12}$$

$$\widehat{UV}_{geo}(x_t) = UV_{geo}(x_s^*) \approx UV_{photo}(x_t). \tag{3.13}$$

Finally, we derive the warped material part mask $M_{g\to p}^{opt}$ and UV map $UV_{g\to p}^{opt}$ for optimization by overlapping regions after warping:

$$M_{g\to p}^{opt} = \widehat{M}_{geo} * M_{photo}, \quad \widehat{M}_{geo}(x_t) = M_{geo}(x_s^*), \tag{3.14}$$

$$UV_{g\to p}^{opt} = \widehat{UV}_{geo} * M_{g\to p}^{opt}. \tag{3.15}$$

Please see Fig. 3.8 for an illustration. In the material optimization stage, $M_{aln}$ refers to $M_{g\to p}^{opt}$.

With the improved view-consistent representation of light sources, we re-optimize the materials to achieve more accurate appearance in the material reoptimization stage. However, the per-pixel lighting bakes-in the geometry in certain views, which necessitates all inputs to be aligned with the geometry. So, we warp the reference photo $I_{photo}$ and the material part mask $M_{photo}$ to match the geometry. We again define bounding box parameters $l_p$ and $c_p$ to compute the warped $\widehat{M}_{photo}$ from $M_{photo}$ to match $M_{geo}$:

$$x_s^* = (x_t - c_g)/l_g * l_p + c_p, \tag{3.16}$$

$$\widehat{M}_{photo}(x_t) = M_{photo}(x_s^*) \approx M_{geo}(x_t), \tag{3.17}$$

$$M_{p\to g}^{opt} = \widehat{M}_{photo} * M_{geo}, \quad I_{p\to g}^{opt} = \widehat{I}_{photo} * M_{p\to g}^{opt}. \tag{3.18}$$

Therefore, $M_{aln}$ refers to $M_{p\to g}^{opt}$ in the material reoptimization stage.

## 3.4 Experiments

### 3.4.1 Datasets

We demonstrate our method on photos and corresponding scene data from several sources and demonstrate its robustness on images and scene geometry of varying quality.

**ScanNet-to-OpenRooms**

We use geometry and 3D part segmentations from OpenRooms [126], corresponding to *multi-view* input images from ScanNet [38] videos, with instance segmentation labels as mask proposals for $M_{photo}$.

**Photos-to-Manual**

For several high-quality real-world photos, we also manually construct matching scenes using Blender [2] from a *single view* and manually segment the material part masks for demonstration purposes.

**SUN-RGBD-to-Total3D**

Our method can also be used for fully automatic material and lighting transfer using a *single-image* mesh reconstruction from Total3D [158] applied to SUN-RGBD [210] inputs. The reconstruction in this case is coarser than CAD retrieval and with a single material per object. We use MaskFormer [34] to obtain segmentation labels as mask proposals and use object classes for mapping.

We assume all meshes have texture coordinates. If not, we use Blender's [2] Smart UV feature to generate them.

### 3.4.2 Material and Lighting Transfer

We demonstrate our material and lighting transfer results in Fig. 3.9, 3.10 and 3.11, where our material prior allows interesting relighting effects such as specular highlights, shadows and


**Figure 3.9.** Example of material transfer results for different scenes with *ScanNet-to-OpenRooms*.

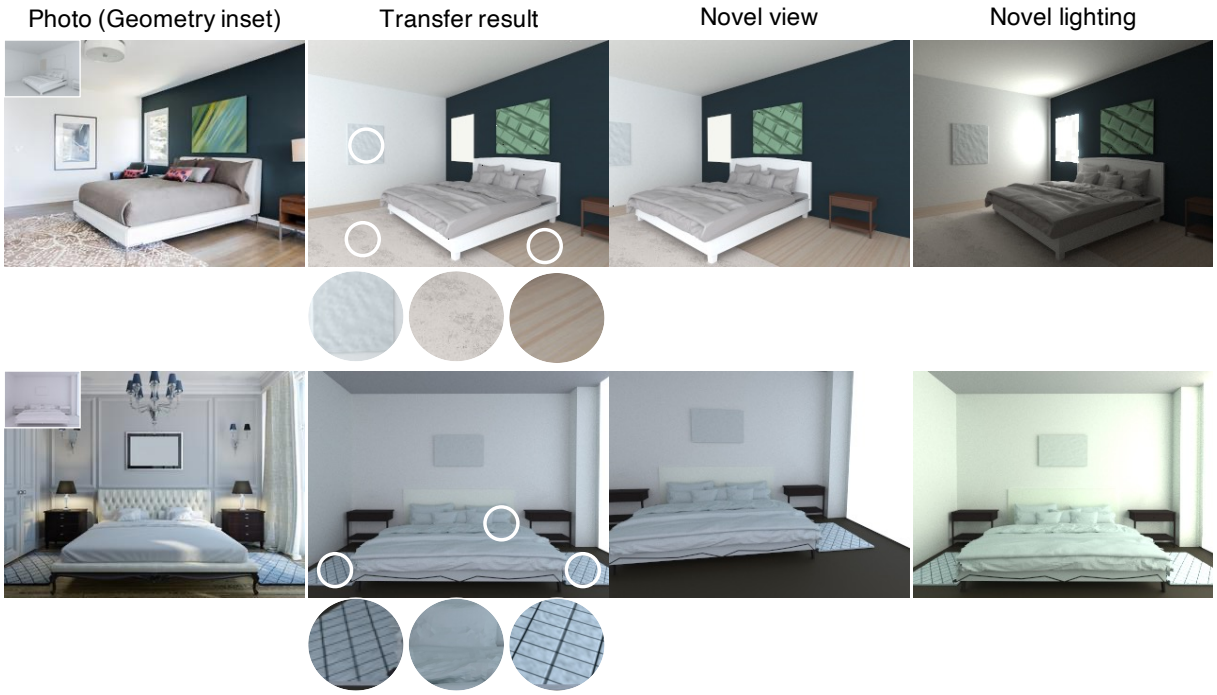



**Figure 3.10.** Our material and lighting transfer results for two scenes in *Photos-to-Manual* dataset. Note how our method is able to accurately reconstruct the appearance and orientation of the spatially-varying materials in these scenes.

global illumination under novel views and lighting.

### Multiview inputs with CAD geometry

We demonstrate the multi-view setup in Fig. 3.9 with *ScanNet-to-OpenRooms* dataset. We select an optimal view for every single material to get the best observation as well as to estimate the global lighting for the entire room. Even though the CAD models are neither perfectly aligned nor perfect replicas of real objects, our method can closely match input appearances. More results are in Sec. 3.4.5.

### Single-image inputs

Our method can even be applied to single image inputs, as shown in Fig. 3.1 and 3.10 with our own *Photos-to-Manual* dataset. Our framework allows material transfer for unseen portions of coarsely aligned CAD models from the photo, as shown in the novel view rendering. This makes


**Figure 3.11.** Examples of material transfer results with *SUN-RGBD-to-Total3D*. Note that these are fully automatic results by using Total3D to reconstruct 3D meshes and panoptic label predictions from MaskFormer. Our method is robust to imperfect geometry and panoptic prediction labels as shown in the inset.


**Figure 3.12.** Example of *editable* variations from originally optimized procedural graph materials for ceiling. We can perturb graph parameters or adjust UV parameters from optimized results to generate various appearances.

the framework more practical than recent works [159] that require perfectly aligned geometry and multiview images to optimize for *observed* geometry. In the second row of Fig. 3.10, both *material* and *orientation* of the carpets with grid patterns are successfully estimated. Lastly, the estimated materials are high-resolution and photorealistic as shown in the zoom-in views and relighting results.

**Automatic 3D reconstruction and masks**

Our method can be fully automatic by using off-the-shelf single image 3D reconstruction and panoptic (or instance) prediction for initialization. We illustrate this in Fig. 3.11 with the *SUN-RGBD-to-Total3D* dataset. This shows that our method is robust even when both masks and meshes are imperfect and not aligned well, which also cannot be achieved in recent works [159] that need high-quality aligned geometry.

**Variations from optimized material**

Another advantage of procedural graph material representation is that it allows further edits from the current parameters. We can adjust material and UV parameters starting from the current estimation and generate edited results as shown in Fig. 3.12. Note that the image becomes brighter by perturbing graph parameters under the same lighting which explains materials can also change the brightness of an image. This demonstrates the benefit of globally consistent lighting optimization with material refinement stages, which ensures materials for each part are consistent under global lighting representation.

### 3.4.3 Baseline Comparisons

**Material classifier**

The most relevant work to ours is PhotoShape [162], which learns a material classifier from a dataset of shapes with material assignments. The input to the network is an image with an aligned material part mask. Although PhotoShape does not consider lighting or complex

**Table 3.1.** User study asking which method produces results more similar to the reference with ScanNet-to-OpenRooms dataset.

| | Classifier | InvRender Med. | Pixel Med. |
|---|---|---|---|
| Ours preferred over | 68.19% | 65.06% | 69.70% |

**Table 3.2.** Similarity evaluation between baselines rendering results and reference photo with ScanNet-to-OpenRooms dataset.

| | Classifier | InvRend. Med. | Pixel Med. | Ours |
|---|---|---|---|---|
| RMSE | 0.452 | 0.349 | 0.337 | **0.259** |
| SSIM | 0.401 | 0.479 | **0.497** | 0.493 |
| LPIPS | 0.546 | 0.510 | 0.501 | **0.489** |

indoor scenes, we compare by mimicking their approach in our setting. We borrow the material classification model from [162] and re-train it in a whole scene setting, with classification of material parts over 886 materials and material category classification over 9 super-classes. For each input image associated with a material part mask, we predict one of 886 material labels. Implementation details can be found in Sec. 3.4.6.

**Median of per-pixel material predictions**

For this baseline, we can construct a homogeneous material from per-pixel predictions of the inverse rendering network [121] by computing median values of per-pixel albedo and roughness under selected view for each material in the masked region and setting the normal to flat.

**Median of pixel values**

We follow IM2CAD [89] to assign a homogeneous albedo as the median values of the 3 color channels independently within the masked region in the selected view for each material, set a fixed roughness value at 0.7 and use a flat normal.

**Comparisons and user study**

The comparisons of our results with baselines are shown in Fig. 3.13 with various datasets.[2] The material classifier can only predict material from a predefined dataset, which is not guaranteed to match the actual appearance in the photo. The median of the inverse rendering method can generate appearances close to the photo, but the albedo color sometimes goes off due to the issue of albedo-lighting ambiguities. The median of photo pixels robustly computes the albedo color similar to the photo, but both the spatially-varying patterns and the roughness are not estimated. In contrast, our method can estimate accurate spatially-varying materials which is similar to the photo as well as the global lighting.

For quantitative evaluations, Table 3.2 reports similarity metrics (RMSE, SSIM, LPIPS) between photos and renderings of various methods with 70 randomly sampled scenes, consisting of 669 material parts, using ScanNet-to-OpenRooms dataset under uniformly sampled views in Table 3.2. We compute RMSE on the optimized region for each material, while SSIM and LPIPS are on the entire image. Note that these similarity metrics are not designed to evaluate similarity between misaligned images or to evaluate spatial variations, so tend to favor homogeneous outputs of the median-based methods. Nevertheless, PhotoScene outperforms all baselines on these metrics, except pixel median in SSIM. Note that the homogeneous albedo from pixel median may match a photo well on an average, but without spatial variations or accurate relighting in new views.

To evaluate methods with human perception, we provide a user study to evaluate the similarity in Table 3.1 using the same dataset. We choose 20 random scenes with uniformly sampled 4 to 12 views and render a set of images under selected views with our result. We ask users on Amazon Mechanical Turk to determine which set of images is more similar to the corresponding photo set. More details can be found in the Sec. 3.4.6. About 65 to 70% users think PhotoScene generates results more similar to the inputs. Thus, our method outperforms the

[2] As none of the baselines can estimate global lighting well, we use our predicted lighting to render baseline images to ensure fair comparison.

**Table 3.3.** Similarity evaluation between rendering results and reference photo on 18 selected scenes of the ScanNet-to-OpenRooms dataset, for baseline methods, various ablations and the full version of the proposed PhotoScene approach.

| | Baseline Methods | | | Ablative Variants of PhotoScene | | | | | | | PhotoScene |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Classifier | InvRend. Med. | Pixel Med. | (a) | (b) | (c) | (d) | (e) | (f) | (g) | Full |
| RMSE | 0.448 | 0.381 | 0.314 | 0.250 | 0.255 | 0.251 | 0.249 | 0.326 | 0.243 | 0.272 | 0.244 |

baselines both qualitatively and quantitatively.

### 3.4.4 Ablation Study

We provide ablation study on our entire framework, by removing each component one at a time. We visualize the final results and compute RMSE on the optimized region for each material part with one scene from Photo-to-Manual dataset as shown in Fig. 3.14 and with 18 randomly selected scenes over 171 materials from ScanNet-to-OpenRooms dataset in Table 3.3. We demonstrate the results a) without warping, b) with random graph selection, c) with material classifier graph selection, d) with stat loss only, e) with VGG loss only, f) without UV transformation parameters, and g) without material reoptimization, as well as our full framework and the baselines.

Without warping, the mask and UV map cannot correctly fetch accurate material regions for optimization. This might lead to wrong material portions being considered due to misalignment so that the overall color and the pattern are not accurate. If we choose procedural graphs randomly from the entire collection or conditioned on a material super-class, the results do not have similar patterns as each procedural graph represents a distinct type of material (e.g. wood, homogeneous, ... etc.). With only statistics loss, the spatially-varying patterns become unconstrained and only match color statistics without considering spatial structures. The UV parameters cannot be estimated correctly and the statistics loss does not contain structure information. With only VGG loss, the results have similar spatial structures but are not guaranteed to have similar color to the reference photo without statistics loss. Without optimization of UV


**Figure 3.13.** Qualitative comparison[2] with baselines on various datasets, where our method generates high-quality materials with spatially-varying patterns that better match the input photograph.

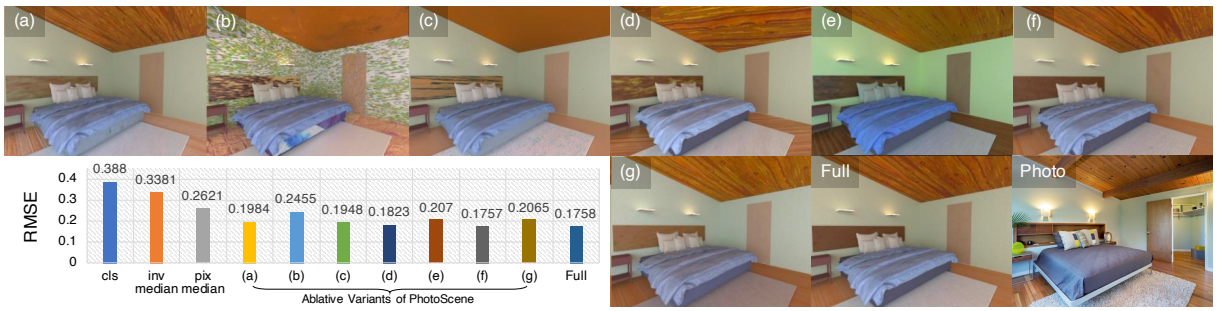



**Figure 3.14.** Ablation study on our entire framework with one selected scene from Photos-to-Manual. We compare the results by removing different modules from our full framework: a) without warping, b) with random graph selection, c) with material classifier graph selection, d) with stat loss only, e) with vgg loss only, f) without UV transformation parameters, and g) without material reoptimization, as well as our full framework and the baselines.

transformation, the orientation and scale of the textures are not guaranteed to be consistent to the reference photo. Note that even though our full method has slightly higher RMSE than (f), its qualitative superiority is not reflected in the metric since our optimization objectives are to align the pixel statistics and masked VGG features rather than per-pixel appearances. Without material re-optimization, sometimes the initial albedo colors have lighting baked-in, resulting in mismatched color under globally-consistent lighting. It is possible to get lower RMSE values with worse UV parameters. In sum, our full framework generates more similar appearances to the photo by considering all the components.

### 3.4.5 More Results

We demonstrate more results on ScanNet-to-OpenRooms, Photos-to-Manual, SUN-RGBD-to-Total3D material and lighting transfer results with novel view and relighting results in Fig. 3.15, 3.16, 3.17, 3.18 and supplementary videos[3].

We also provide results with panoptic labels predicted by MaskFormer [34] instead of ground truth labels for ScanNet-to-Openrooms, and compute the results with baselines with randomly selected 62 scenes and over more than 521 materials, as shown in Table 3.4. The RMSE errors are slightly higher when using panoptic predictions, but still lower than baseline

[3]Videos can be found on project page: https://yuyingyeh.github.io/projects/photoscene.html

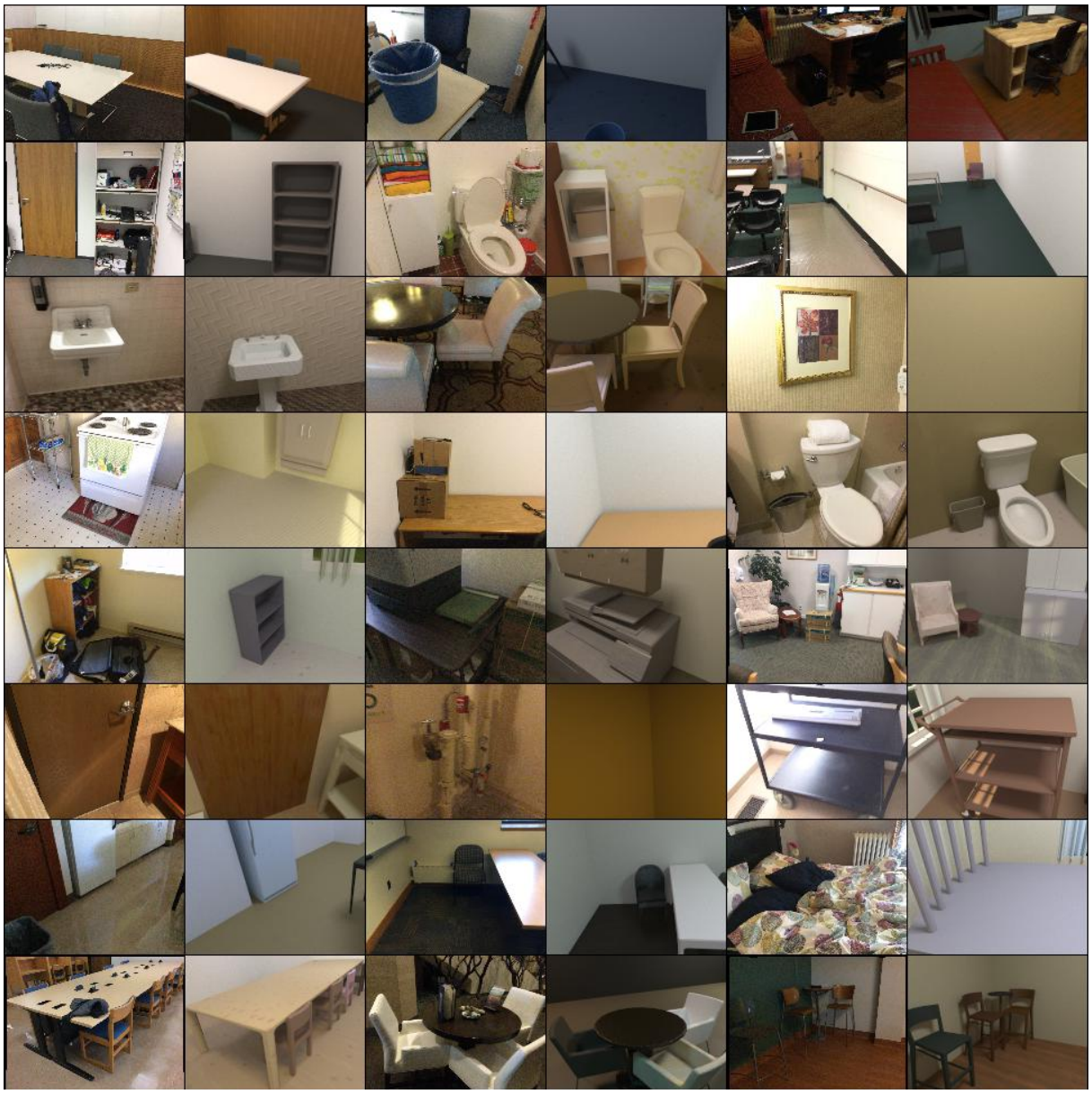


**Figure 3.15.** More results of material and lighting transfer with ScanNet-to-OpenRooms dataset.

methods with panoptic ground truths. This demonstrate that our method is robust to imperfect input mask and outperform baseline methods regardless the input masks.

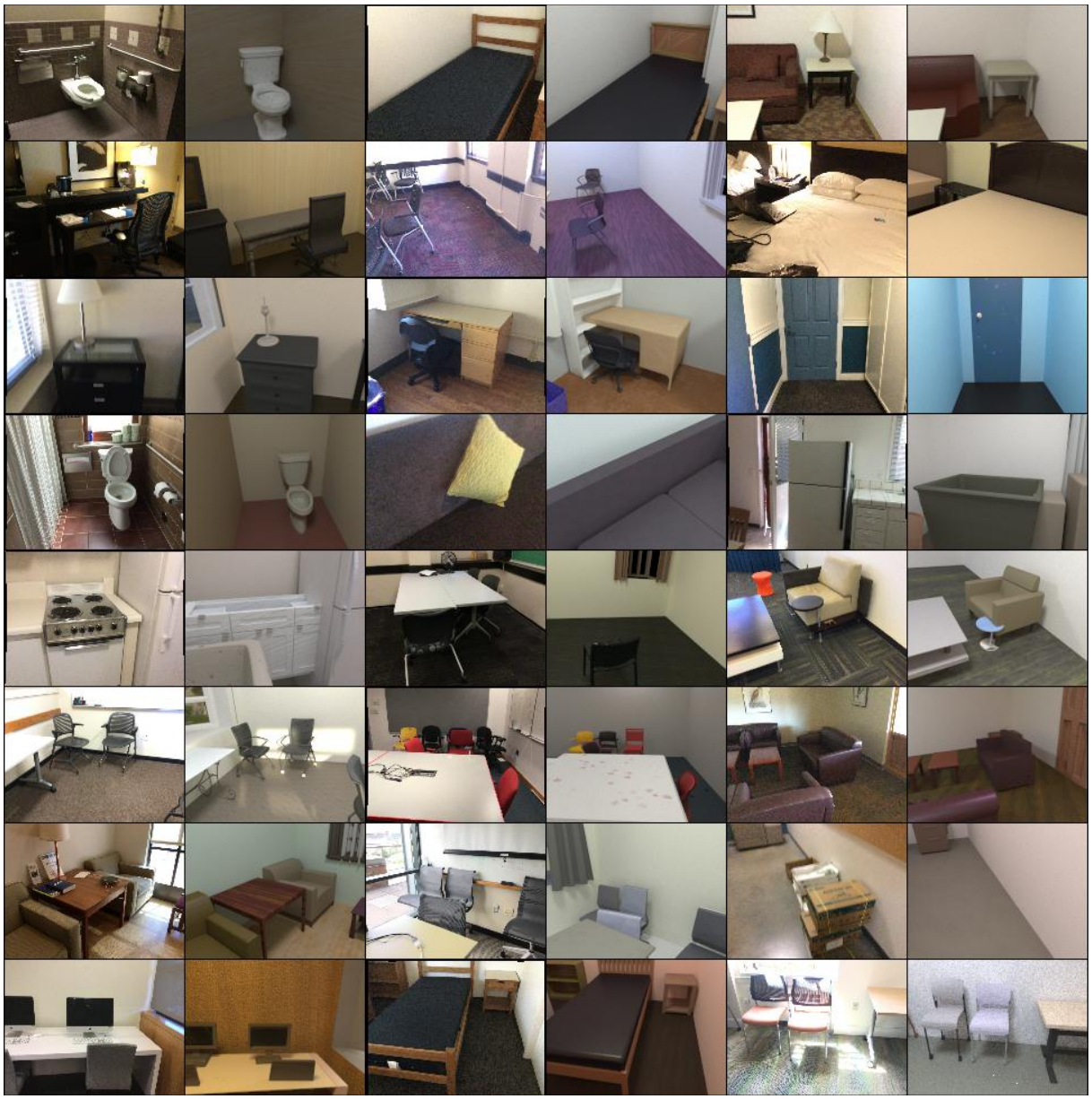


**Figure 3.16.** More results of material and lighting transfer with ScanNet-to-OpenRooms dataset.

### 3.4.6 Additional Details for Experiments

#### Material classifier implementation

The material classification model is based on ResNet-18 [73] backbone. We represent 2D convolution by *Conv2D*(C, K, S, P) where C is the output channels, K is the kernel size, S is stride and P is padding. Other operations in the model include *BN* for 2D batch normalization,

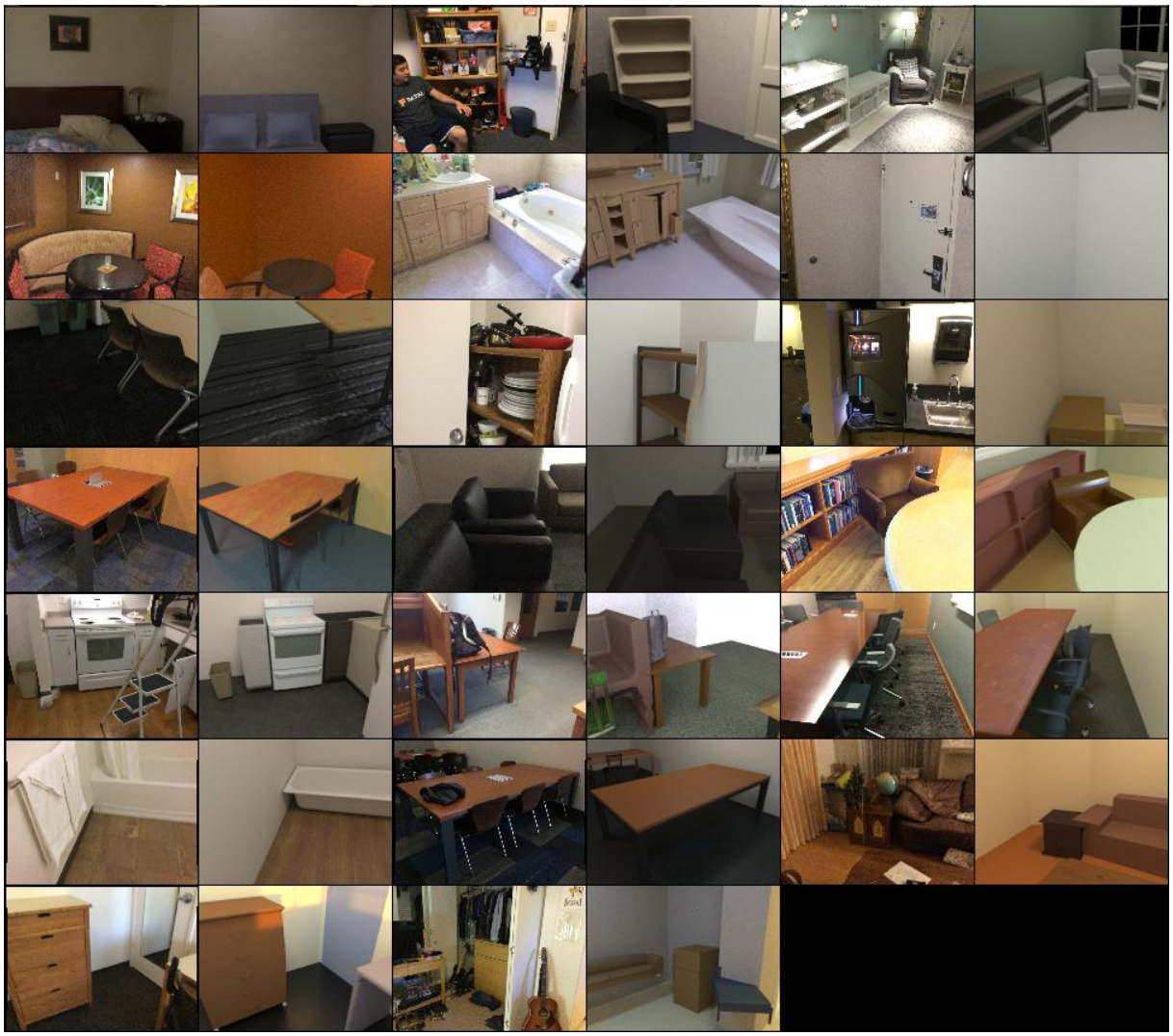


**Figure 3.17.** More results of material and lighting transfer with ScanNet-to-OpenRooms dataset.

**Figure 3.18.** More results of material and lighting transfer with SUN-RGBD-to-Total3D dataset.

ReLU, and Maxpool(K, S) for 2D max-pooling of kernel size K and stride S. The model takes the concatenation of the image and a binary mask of size 240×320×4 as input, followed by *Conv2D*(64, 7, 2, 3), BN, ReLU, Maxpool(3, 2), and modules *conv2.x*, *conv3.x*, *conv4.x*, *conv5.x*

**Table 3.4.** Similarity evaluation between baselines rendering results and reference photo with ScanNet-to-OpenRooms dataset using ground truth panoptic labels versus predictions from MaskFormer [34].

| | Classifier | InvRend. Med. | Pixel Med. | Ours |
|---|---|---|---|---|
| RMSE (GT Mask) | 0.453 | 0.337 | 0.342 | 0.259 |
| RMSE (Pred. Mask) | 0.467 | 0.373 | 0.354 | 0.285 |

from ResNet-18, and 2D average-pooling, resulting by a feature vector of dimension 512. With the feature vector as input, a fully-connected (FC) layer classifies over 886 bins of materials and another FC layer classifies over 9 super-classes. A standard cross-entropy loss is used for the classification heads.

**User study details**

There are 60 random AMT users; each is asked to make a different binary comparison for each of 20 scenes *without* pre-training on our task. In each comparison, we ask users to choose the better set of multi-view renderings of transfer results between ours and one randomly sampled baseline from {cls, inv med, pix med}. Two options are randomly placed while the input photo is in the middle. Each comparison is evaluated by 20 different users.

### 3.4.7 Discussion and limitations.

Our algorithm assumes a part segmentation already exists in the reconstructed geometry, and our results depend on its quality and granularity. A finer part segmentation could be achieved by retrieving objects from a higher quality CAD model collection. Our graph collection is limited to the set provided by the existing implementation of MATch [200], though more general procedural graphs could be added with some effort, possibly using automatic techniques [84]. Our approach cannot handle specific patterns such as paintings, which could be addressed by training a generative model for such materials [67]. Lastly, we rely on a neural inverse rendering initialization, where the current state-of-the-art is restricted to small resolutions, so

some high-frequency information from photos might be lost. This will likely be improved by future architectures handling higher resolutions.

Our approach can synthesize high-quality digital counterparts of real scenes which may be rendered to create photorealistic images. Using a physically-based material prior also allows the ability to edit properties of these images by generating plausible new materials for specific regions or objects, which may be used for potentially harmful purposes. An avenue to overcome this negative impact might be further research in digital watermarks such as [220] for materials generated through our material priors, embedded in a manner that allows them to persist in an identifiable way through the rendering process.

## 3.5 Conclusion

We have presented a novel approach to transfer materials and lighting to indoor scene geometries, such that their rendered appearance matches one or more input images. Unlike previous work on material transfer for objects, we must handle the complex inter-dependence of material with spatially-varying lighting that encodes distant interactions. We achieve this through an optimization that constrains the material to lie on an SVBRDF manifold represented by procedural graphs, while solving for the material parameters and globally-consistent lighting with a differentiable renderer that best approximates the image appearances. We demonstrate high-quality material transfer on several real scenes from the ScanNet, SUN-RGBD dataset and unconstrained photographs of indoor scenes. Since we estimate tileable materials that can be procedurally generated, the scenes with transferred material can be viewed from novel vantage points, or under different illumination conditions, while maintaining a high degree of photorealism. We believe our work may have significant benefits for 3D content generation in artistic editing and mixed reality applications. Further, our approach can be used to create datasets for inverse rendering, where geometry is easier to acquire but ground truth material and lighting are hard to obtain.

Chapter 3 is based on the material as it appears in IEEE Conference on Computer Vision

and Pattern Recognition (CVPR), 2022 (”PhotoScene: Photorealistic Material and Lighting Transfer for Indoor Scenes”, Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, Manmohan Chandraker). The dissertation author was the primary investigator and author of this paper.

# Chapter 4

# Image-guided Texture Synthesis through Geometry-aware Diffusion

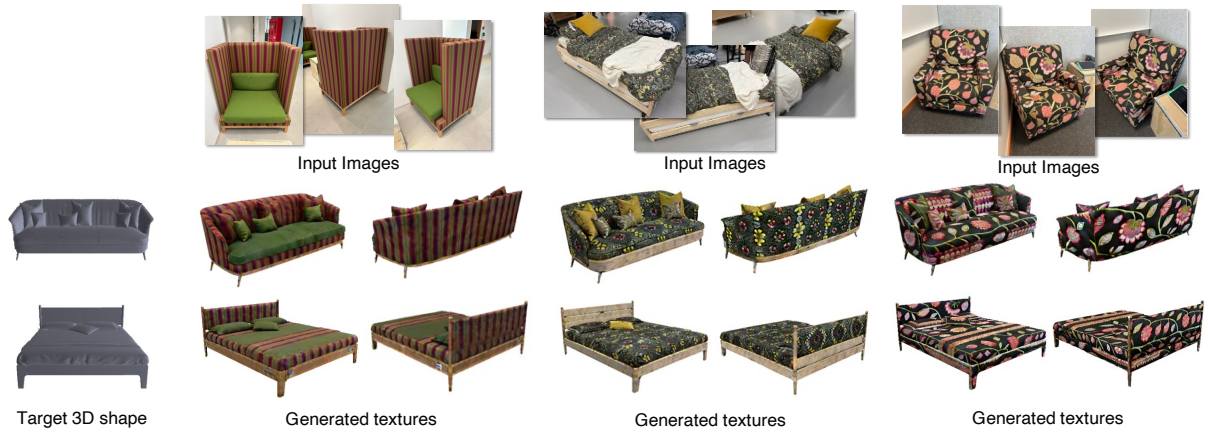



**Figure 4.1.** Texture transfer from sparse images. Given a small number of images and a target mesh, our method synthesizes geometry-aware texture that looks similar to the input appearances for diverse objects.

## 4.1 Introduction

High-quality 3D content is indispensable for a wide range of critical applications, including AR/VR, robotics, film, and gaming. In recent years, remarkable progress has been made in democratizing 3D content creation pipelines, facilitated by advancements in 3D reconstruction [145, 152] and generative models [62, 209]. While substantial attention has been devoted to exploring the *geometry component* [245, 35, 24] and neural implicit representations [161], such as NeRF [145], creation of high-quality *textures* is relatively under-explored. Textures are pivotal

in creating realistic, highly detailed appearances and are integral to various graphics pipelines, where industry has traditionally relied on professional, experienced artists to craft textures. This process usually involves manually authoring procedural graphs [4] and UV maps, making it expensive and inefficient. Automatically transferring the diverse visual appearance of objects around us to the texture of any target geometry would thus be highly beneficial.

We present *TextureDreamer*, a novel framework to create high-quality relightable textures from sparse images. Given 3 to 5 randomly sampled views of an object, we can transfer its texture to an target geometry that may come from a different category. This is an extremely challenging problem, as previous texture creation methods usually either require densely sampled views with aligned geometry [19, 273, 116], or can only work for category-specific shapes [207, 20, 166, 74]. Our framework draws inspiration from recent advancements in diffusion-based generative models [209, 211, 77]. Trained on billions of text-image pairs, these diffusion models enable text-guided image generation with extraordinary visual quality and diversity [181]. Pioneering works have applied these pre-trained 2D diffusion models to text-guided 3D content creation [171, 127, 233]. However, a common limitation among those methods is that *text-only input* may not be sufficiently expressive to describe complex, detailed patterns, as demonstrated in Figure 4.2. In contrast to text-guided methods, we effectively extract texture information from a small set of input images by fine-tuning the pre-trained diffusion model with a unique text token [54, 188]. Our framework, therefore, addresses the challenge of accurately describing complex textures.

The Score Distillation Sampling (SDS) [171, 229] is one core element that bridges pre-trained 2D diffusion models with 3D content creation. It is widely used to generate and edit 3D contents by minimizing the discrepancy between the distribution of rendered images and the distribution defined by the pre-trained diffusion models [127, 141]. Despite its popularity, two well-known limitations impede its ability to generate high-quality textures. First, it tends to create over-smoothed and saturated appearances due to the unusually high classifier-free guidance necessary for the method to converge. Second, it lacks the knowledge to generate

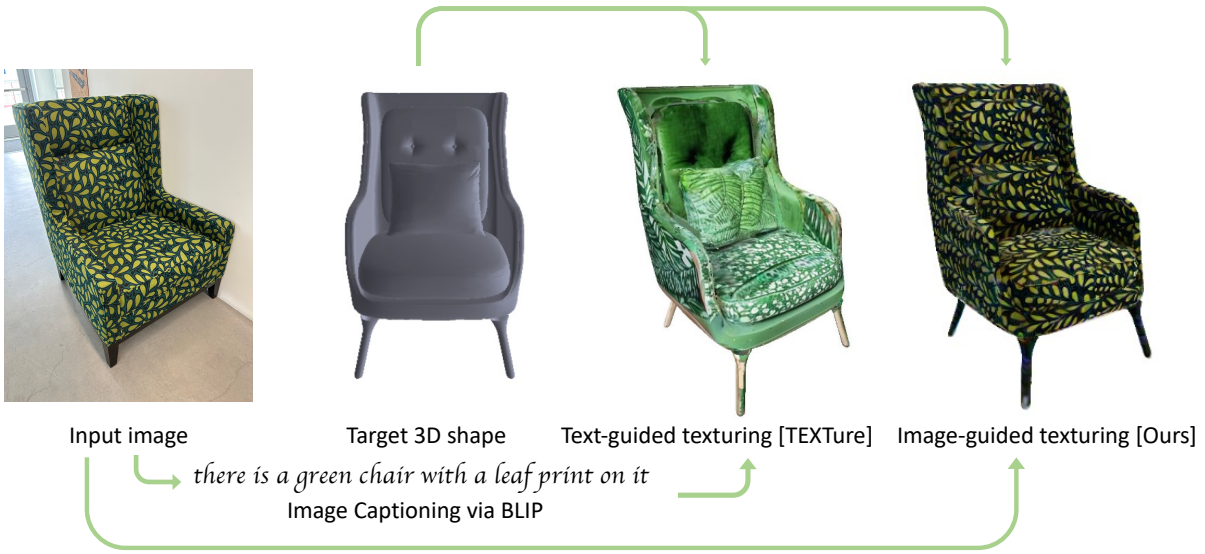



**Figure 4.2.** Limitation of text-guided texturing. Compared to text-guided texturing method which requires a captioning method to generate a text prompt which might not express all the details of the image, image-based guided texturing can be more effective and more expressive. Image captioning is predicted by BLIP [118], text-guided texturing is generated via TEXTure [183], and image-guided result is from our method.

a 3D-consistent appearance, often resulting in multi-face artifacts and mismatches between textures and geometry.

We propose two key design choices to tackle these challenges. Instead of using SDS, we build upon Variational Score Distillation (VSD) in our optimization approach, which can generate much more photorealistic and diverse textures. Initially introduced in ProlificDreamer [233], VSD treats the whole 3D representation as a random variable and aligns its distribution with the pre-trained diffusion model. It does not need a large classifier-free guidance weight to converge, which is essential to create a realistic and diverse appearance. However, naïvely applying VSD update does not suffice for generating high-quality textures in our application. We identify a simple modification that can improve texture quality while slightly reducing the computational cost. Additionally, VSD alone cannot fully solve the 3D consistency issue. Fine-tuning on sparse inputs makes converging harder, as observed by previous work [179]. We, therefore, explicitly condition our texture generation process on geometry information extracted from the given mesh by injecting rendered normal maps into the fine-tuned diffusion model through the

ControlNet [264] architecture. Our framework, designated as personalized geometry aware score distillation (PGSD), can effectively transfer highly detailed textures to diverse geometry in a semantically meaningful and visually appealing manner. Extensive qualitative and quantitative experiments demonstrate that our framework substantially outperforms state-of-the-art texture-transfer methods.

## 4.2 Related Works

### 4.2.1 Texture Synthesis and Reconstruction

Classical texture creation methods involve sampling from a distribution derived from the neighborhood [49, 108], tiling repetitive patterns [112, 186], or with generative approaches [149, 274]. These methods fall short in creating semantic meaningful textures. Texture reconstruction by fusing multi-view images onto the object surfaces [19, 273, 116] requires highly accurate geometry reconstruction. Numerous learning-based methods were proposed to learn texture creation from large-scale 3D datasets [33, 20, 207, 74, 166] but are confined to specific categories within the dataset. Recent works also use CLIP model [178] for text-guided texture generation of arbitrary objects [143, 32, 148, 134], but their texture qualities are usually low. In contrast, TextureDreamer can create semantically meaningful, high-quality textures for arbitrary objects using uncorrelated sparse images. Traditionally, textures are represented as a 2D image and projected to object surfaces through UV mapping. Leveraging the recent progress in neural implicit representation, our method, along with recent developments in inverse rendering [57, 23, 21, 214] and 3D generation [57, 23], represents texture as a neural implicit texture field.

### 4.2.2 Diffusion Models

Diffusion models [209] have emerged as the state-of-the-art generative models [77, 211], demonstrating exceptional visual quality [181]. Its training and inference involve iteratively adding noise with different variances and denoise the data. Trained on internet-scale image-text

pair datasets [181], these pre-trained models exhibit unprecedented capability in text-guided image synthesis and have proven successful in various image editing tasks. Recent works also manage to fine-tune pre-trained diffusion models on much smaller datasets or even a few images to facilitate customized/personalized image synthesis [188] and image generation conditioned on multi-modal data [264], such as normal and semantic maps. Building upon this progress, TextureDreamer can effectively extract texture information from sparse views and transfer it to a novel target object in a geometry-aware manner.

### 4.2.3 3D Generation with 2D Diffusion Priors

Diffusion-based 3D content creation has very recently gained substantial interest. Several methods directly train 3D diffusion models to generate 3D content in various representations, including point cloud [133], neural radiance filed [97], hyper-network [50] and texture [260]. Others utilize pre-trained 2D diffusion models by either progressively fusing generated images from different views [183, 22, 27, 6] or optimizing the 3D representation through score distillation sampling [127, 141, 171] and its improved variations [101, 233]. While many methods concentrate on text-guided 3D generation, fewer attempt to leverage diffusion models to generate 3D content from images. A number of concurrent works fine-tune 2D diffusion models on large-scale 3D datasets for sparse view reconstruction [173, 202], primarily focusing on whole 3D object reconstruction. In contrast, TextureDreamer targets transferring textures from a small number of images to a target 3D shape with unmatched geometry. Dreambooth3D [179] and TEXTure [183] extract information from sparse views into a new text token and fine-tuned diffusion model weights, which can be used to generate personalized 3D object or texture unseen objects. TextureDreamer employs a similar method to extract information from sparse images. However, it differs from prior works on utilizing the extracted information for texture generation, leading to improvements in consistency and photorealism.

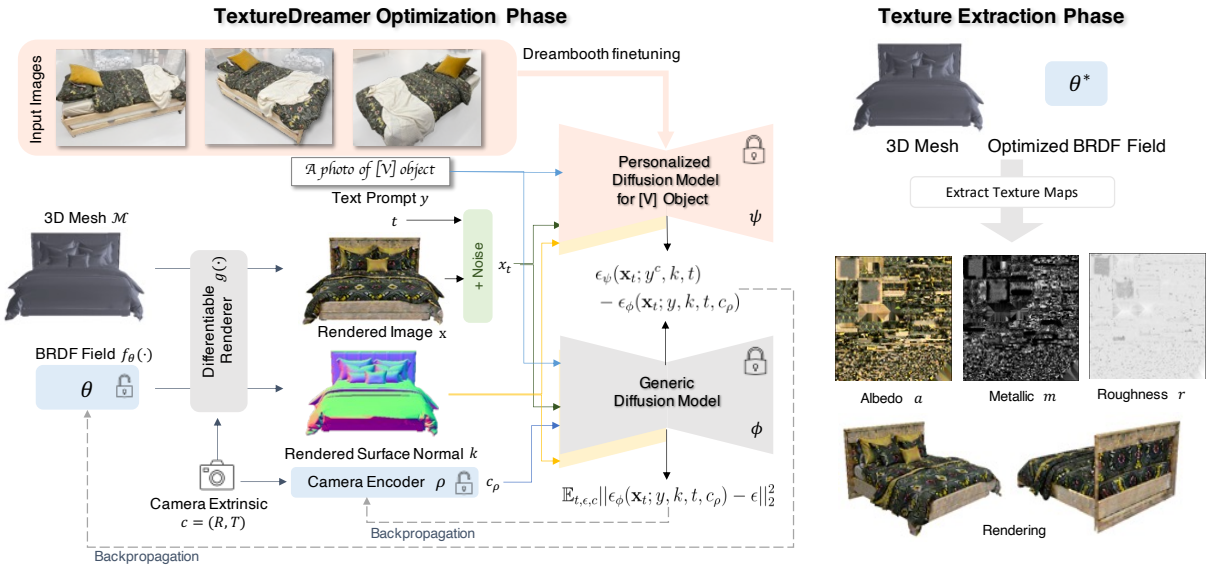



**Figure 4.3. Overview of TextureDreamer**, a framework which synthesizes texture for a given mesh with appearance similar to 3-5 input images of an object. We first obtain personalized diffusion model $\psi$ with Dreambooth [188] finetuning on input images. The spatially-varying bidirectional reflectance distribution (BRDF) field $f_\theta$ for the 3D mesh $\mathscr{M}$ is then optimized through personalized geometric-aware score distillation (PGSD) (detailed in Section 4.3.2). After optimization finished, high-resolution texture maps corresponding to albedo, metallic, and roughness can be extracted from the optimized BRDF field.

## 4.3 Method

We propose TextureDreamer, a framework which synthesizes geometry-aware texture for a given mesh with appearance similar to 3-5 input images of an object. In Section 4.3.1, we first introduce preliminaries on Dreambooth [188], ControlNet [264] and score distillation sampling [171, 229, 233]. In Section 4.3.2, we propose personalized geometry-aware score distillation (PGSD), which is our core technical contribution that enables high-quality image-guided texture transfer from sparse images to arbitrary geometries.

### 4.3.1 Preliminaries

**Dreambooth** [188] is a simple yet effective method to fine-tune pre-trained text-to-image diffusion models on a small number of input images for personalized text-guided image generation. It stores the subject's appearance into the diffusion model weights with a specific text-

token *"[V]"*. Dreambooth is fine-tuned with two loss functions. Reconstruction loss is standard diffusion denoising supervision on the input images. Class-specific prior preservation loss is proposed to avoid language drift and loss of diversity caused by fine-tuning. It further supervises the pre-trained model with a large number of its own generated examples. TextureDreamer uses DreamBooth to distill texture information from input images. Instead of image synthesis, we apply the distilled information to a 3D object with different geometry.

**ControlNet** [264] proposes a novel architecture that adds spatial conditioning control to pre-trained diffusion models. The key insight is to reuse the large number of diffusion model parameters trained on billions of images and insert small convolution networks into the model with window size 1 and zero-initialized weights. It enables robust fine-tuning performance on small datasets with different types of 2D conditions, such as depth, normal, and edge maps. We utilize ControlNet models to ensure that our created textures are aligned with the given geometry.

**Score Distillation Sampling** [171, 229] is the core component of numerous methods that use pre-trained 2D diffusion models for 3D content creation [171, 127, 30]. It optimizes the 3D representation by pushing its rendered images to a high-dimensional manifold modeled by the pre-trained diffusion model. Let $\theta$ be the 3D representation and $\varepsilon_\psi$ be the pre-trained diffusion model. The gradient back-propagated to the parameter $\theta$ is

$$\nabla_\theta \mathscr{L}_{\text{SDS}}(\theta) \triangleq \mathbb{E}_{t,\varepsilon}\left[w(t)(\varepsilon_\psi(\mathbf{x}_t, y^c, t) - \varepsilon)\frac{\partial g(\theta, c)}{\partial \theta}\right],$$

where $\omega(t)$ is the weight coefficient, $y$ is the text input, $y^c$ denotes view-dependent conditioning, $t$ is the time step, $c$ is the camera pose, $g(\cdot)$ is a differentiable renderer, $\mathbf{x}_t$ is the noisy image computed by adding noise to the rendered image $\mathbf{x} = g(\theta, c)$ with variance dependent on time $t$. Despite its wide usage, SDS requires a much higher weight than normal classifier-free guidance [78] to converge, oversmoothed and oversaturated appearance. To overcome this issue, Wang et al. [233] propose an improved version, called variational score distillation (VSD), which can converge with standard classifier-free guidance. VSD treats the whole 3D representation $\theta$

as a random variable and minimizes the KL divergence between $\theta$ and the distribution defined by the pre-trained diffusion model. It involves fine-tuning a LoRA [82] network $\varepsilon_\phi$ (and a camera encoder $\rho$ which embeds camera pose $c$ as an condition input to $\varepsilon_\phi$) to denoise the noisy images generated from 3D representation $\theta$

$$\min_{\phi} \mathbb{E}_{t,\varepsilon,c}\left[||\varepsilon_\phi(\mathbf{x}_t, y, t, c) - \varepsilon||_2^2\right] \tag{4.1}$$

The gradient to the 3D representation $\theta$ is then computed as

$$\mathbb{E}_{t,\varepsilon,c}\left[w(t)(\varepsilon_\psi\left(\mathbf{x}_t, y^c, t\right) - \varepsilon_\phi(\mathbf{x}_t, y, t, c)\right)\frac{\partial g(\theta, c)}{\partial \theta}\right]. \tag{4.2}$$

While VSD significantly improves both visual quality and diversity of generated 3D contents, it cannot address the 3D consistency issue due to the inherent lack of 3D knowledge, leading to multi-face errors and mismatches between geometry and textures. We address this challenge by explicitly injecting geometry information to make our diffusion model geometry aware.

### 4.3.2 Personalized Geometry-aware Score Distillation (PGSD)

**Problem setup**

We illustrate our method in Figure 4.3. The inputs to our framework include a small set of images (3 to 5) casually captured from different views $\{I\}_{k=1}^{K}$ and a target 3D mesh $\mathscr{M}$. The outputs of our framework are relightable textures transferred from image set $\{I\}_{k=1}^{K}$ to $\mathscr{M}$ in a semantically meaningful and visually pleasing manner. Our relightable textures are parameterized as standard microfacet bidirectional reflectance distribution (BRDF) model [96], which consists of 3 parameters, diffuse albedo $a$, roughness $r$, and metallic $m$. We deliberately *do not* optimize normal maps as it encourages the pipeline to fake details that are inconsistent with mesh $\mathscr{M}$. Following the recent trend of neural implicit representation [152, 154, 72], during optimization, we represent our texture as a neural BRDF field $f_\theta(v) : v \in \mathbb{R}^3 \rightarrow, a, r, m \in \mathbb{R}^5$,

where $v$ is an arbitrary point sampled on the surface of $\mathscr{M}$ and $f_\theta$ consists of a multi-scale hash encoding and a small MLP. We find such an implicit representation can better regularize the optimization process, leading to smoother textures. However, given the UV mapping of $\mathscr{M}$, our representation can also be converted to standard 2D texture maps that are compatible with standard graphics pipelines, by querying every 3D point corresponding to each texel, as shown on the right-hand side of Figure 4.3.

**Personalized texture information extraction**

We follow Dreambooth [188] to extract texture information from sparse images. To be specific, we fine-tune a personalized diffusion model on input images with a text prompt $y$, *"A photo of [V] object"*, where *"[V]"* is a unique identifier to describe the input object. Compared to the alternative textual inversion method [54], we observe that Dreambooth converges faster and can better preserve intricate texture patterns, possibly due to its larger capacity. We first mask out the background of the target object with a white color. For the reconstruction loss, we resize the shorter edge of input images to 512 and randomly crop 512x512 patches for training. We do not apply class-specific prior preservation loss, as we hope our Dreambooth finetuning model can generalize to other categories. We also experiment with different variations, including jointly fine-tuning the text encoder and replacing the diffusion denoising network with a pre-trained ControlNet, but do not observe any improvements.

**Geometry-aware score distillation**

Once we finish extracting texture information with Dreambooth, we transfer the information to mesh $\mathscr{M}$ by adopting the fine-tuned Dreambooth model as the denoising network $\varepsilon_\psi$ for score distillation sampling. Specifically, we choose VSD instead of the original SDS because of its superior ability to generate highly realistic and diverse appearances. To render images $\mathbf{x}$ for VSD gradient computation, we follow Fantasia3D [30] to pre-select a fixed HDR environment map $E$ as illumination and use Nvdiffrast [114] as our differentiable renderer. We

set the object background to be a constant white color to match the input images for Dreambooth training. We observe this can help achieve better color fidelity compared to random color or neutral background.

However, simply replacing SDS with VSD cannot address the limitation of lacking 3D knowledge in 2D diffusion models. We thus propose geometry-aware score distillation, where we inject geometry information extracted from mesh $\mathscr{M}$ into our personalized diffusion model $\varepsilon_\psi$ through a pre-trained ControlNet conditioned on normal maps $k$ rendered from $\mathscr{M}$. This augmentation significantly boosts 3D consistency of generated textures (see Figure 4.7). With the ControlNet conditioning, the pillow texture from the input images can be accurately matched to the target shape, despite the shape mismatch. We experiment with different ControlNet conditions and show that normal conditions can best prevent texture-geometry mismatch.

Let $\mathbf{x} = g(\theta, c)$ be the rendered image under a fixed environment map from camera pose $c$ with the extracted BRDF maps $a_\theta, r_\theta, m_\theta$. The gradient of proposed Personalized Geometry-aware Score Distillation (PGSD) to optimize the MLP parameter $\theta$ of BRDF field is:

$$\begin{aligned}&\nabla_\theta \mathscr{L}_{\text{PGSD}}(\theta)\\ &\quad\triangleq \mathbb{E}_{t,\varepsilon,c}[w(t)(\varepsilon_\psi(\mathbf{x}_t; y^c, k, t) - \varepsilon_\phi(\mathbf{x}_t; y, k, t, c_\rho))\frac{\partial \mathbf{x}}{\partial \theta}],\end{aligned}$$

where $\mathbf{x}_t = \alpha_t \mathbf{x} + \sigma_t \varepsilon$ is the rendered image $\mathbf{x}$ perturbed by noise $\varepsilon \sim \mathscr{N}(\mathbf{0}, \mathbf{I})$ at time $t$, $c_\rho$ is the embedding of the camera extrinsic $c$ encoded by a learnable camera encoder $\rho$, $\varepsilon_\psi$ and $\varepsilon_\phi$ are the fine-tuned personalized diffusion model and the generic diffusion model pretrained on a large-scale dataset, respectively. Both models are augmented with ControlNet conditioned on normal map $k$, as shown in the yellow part underneath the diffusion model in Figure 4.3.

We found that our method does not benefit from classifier-free guidance (CFG) [78], probably because the personalized model $\varepsilon_\psi$ has been fine-tuned on a small number of images. Since our goal is to faithfully transfer input appearance to target shape, it is not necessary to have CFG to increase the diversity. Similar observation can be found in recent literature [199].

We additionally identify several important design choices through extensive experiments. First, it is important to initialize the $\varepsilon_\phi$ in Eq. 4.1 with original pre-trained diffusion model weights while the Dreambooth weight will remove texture details. This is probably because the Dreambooth fine-tuning process makes the diffusion model overfit to a small training set, as pointed out by previous work [179]. Moreover, we find that removing the LoRA weights can substantially improve texture fidelity. Similar difficulties in training LoRA were also reported in [201]. We therefore implement our personalized geometry-aware score distillation loss $\mathscr{L}_{PGSD}$ by removing the LoRA structure in $\varepsilon_\phi$ and only keeping the camera embedding, achieving the best quality. We show more comparisons in Figure 4.7.

## 4.4 Experiment

### 4.4.1 Experimental Setup

#### Dataset

We conduct our experiments on 4 categories of objects: sofa, bed, mug/bowl, and plush toy. For each category, we select 8 instances of objects and create a small image set by casually sampling 3 to 5 views surrounding the object, resulting in 32 image sets in total. For every image in the 32 image sets, we apply U2-Net [177] to obtain the foreground mask automatically or use a semi-auto background removal application[1] to obtain more accurate masks. We perform texture transfer for each image set to diverse meshes including but not limited to same category shapes, different category shapes, or even geometry with different genus numbers. To test our texture-transferring framework, we select 3 meshes for each of the 4 categories that are dissimilar to the captured image sets. We acquire these 3D meshes from 3D-FUTURE [53] and online repositories.[2][3]. We run intra-class texture transfer for all 4 categories of objects and also run inter-class texture transfer between bed and chair, to test our method's generalization ability.

[1] https://www.remove.bg/upload
[2] https://www.cgtrader.com/
[3] https://sketchfab.com/


**Figure 4.4.** Image-guided transfer results from four categories (beds, sofas, plush toys, and mugs) of image sets to diverse objects. Our method can be applied to various object types and transfer the textures to diverse object shapes.

**Figure 4.5.** Example of cross-category texture synthesis results. Input images (top row) can guide the texture synthesis (bottom row) for shapes which are not in the same category.

#### Implementation details

We implement our framework based on PyTorch [164] and Threestudio [68]. We use latent diffusion and ControlNet v1.1 as our pre-trained diffusion model and ControlNet respectively. In all our experiments, we set the classifier-free guidance weight of $\mathscr{L}_{PGSD}$ as 1.0 (equivalent to setting $\omega = 0$ in the original CFG formulation). Following DreamFusion [171], we also apply view-dependent conditioning to the input text prompt. The BRDF field is parameterized with an MLP using hash-grid positional encoding [152], following prior works [30, 233]. Our camera encoder consists of two linear layers that project the camera extrinsic to a latent vector of $1,280$ dimensions to be fused with time and text embedding in U-Net. We empirically set the learning rate to 0.01 for encoding, 0.001 for MLP, and 0.0001 for camera encoder for all experiments.

### 4.4.2 Baseline Methods

Latent-paint [141] and TEXTure [183] are two recent text-guided texturing methods with 2D diffusion prior. They also demonstrate the capability of texturing meshes from images. Latent-paint leverages the Texture Inversion [54] to extract image information into text embedding and distills the texture with SDS. TEXTure first finetunes the pre-trained diffusion model by combining Texture Inversion and Dreambooth [188] and use this fine-tuned model to synthesize texture with an iterative mesh painting algorithm. Following TEXTure, we augment the input images with a random color background. We closely follow the original implementation of baseline methods to run the experiments.

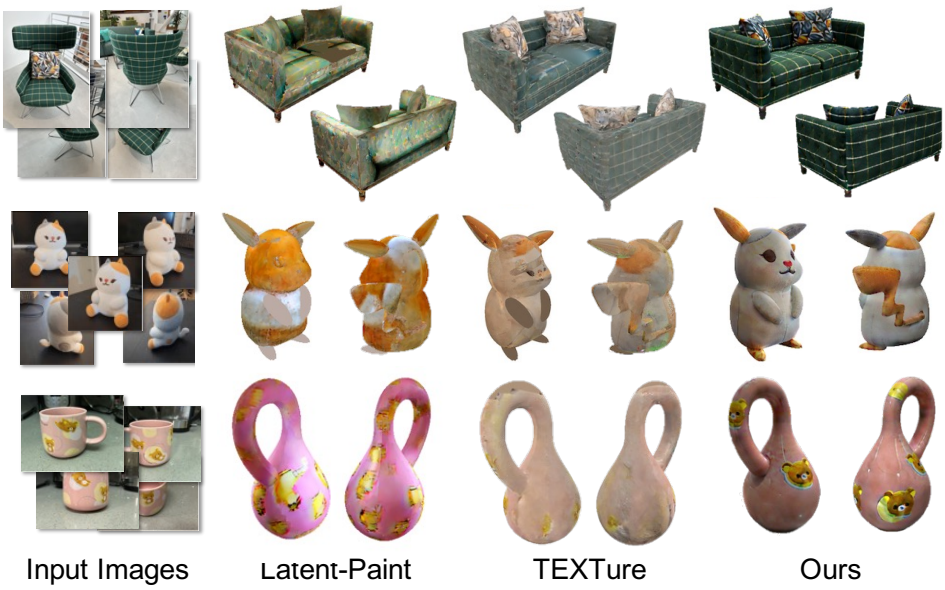



**Figure 4.6.** Comparison between baseline methods. Compared with Latent-Paint [141] and TEXTure [183], our method can synthesize seamless and geometry-aware textures that are compatible with the target mesh geometry.

**Table 4.1.** User study on image-guided texture transfer.

| | Ours preferred over | |
|---|---|---|
| | Latent-Paint | TEXTure |
| Image Fidelity | 71.82% | 69.43% |
| Texture Photorealism | 77.03% | 85.52% |
| Shape-Texture Consistency | 78.49% | 85.16% |

### 4.4.3 Image-guided Texture Transfer

**Qualitative evaluation**

Our method can synthesize geometry-aware and seamless textures that has similar patterns and styles as the input for diverse object geometry. In Figure 4.4, textures can be synthesized for the *same* category. It can be diverse under different seeds as shown in Figure 4.8. We also demonstrate that our method can synthesize textures *across different categories*. In Figure 4.1, we show texture synthesis results from images of sofa to bed shapes, and vice versa. Our method is also capable of performing texture synthesis across a broader range of different categories. In Figure 4.5, high-quality and realistic textures can be synthesized across chair, mug, plush toy,

**Table 4.2.** Quantitative evaluation on image-guided texturing.

| | CLIP similarity ↑ |
|---|---|
| Latent-Paint [141] | 0.7969 |
| TEXTure [183] | 0.7988 |
| Ours | **0.8296** |


**Figure 4.7.** Ablation study. With ControlNet conditioned on normal maps, the result has the best texture-geometry consistency. Without ControlNet or with depth-based ControlNet, the results suffer from texture-geometry misalignment. Using SDS loss leads to blurry textures. Without the LoRA module removed, the results tend to remove the existing texture from the personalized diffusion model. Our full method can synthesize accurate texture which is similar to input appearances.

or even non-rigid objects such as bags or clothes. It can also be used to synthesize texture for shapes captured from a commercial 3D scanner [125], as shown in Figure 4.9.

In Figure 4.6, we qualitatively compare our method with baselines. Two views are shown in each example. Latent-Paint tends to generate textures with colors and patterns that are different from input images. TEXTure can preserve the color and texture better than Latent-Paint, but the texture contains visible seams (possibly due to the iterative painting). Our method can reason the semantics of the geometry (*e.g.* the positions of eyes) and demonstrate higher quality, seamless, and geometry-aware texturing results with higher fidelity from the input images.

**Quantitative evaluation**

It is non-trivial to perform quantitative comparisons for texture transfer due to the shape difference between geometry and photos. We perform a user study to evaluate transfer fidelity, texture photorealism, and texture-geometry compatibility across baselines by asking users the following questions: 1) Which one has the texture that looks more similar to input images? 2) Which one has a texture which looks more like a real object? 3) Which one has the texture which is more compatible with the meshes? (Which texture painted more fitted to the geometry?) We conduct a user study with Amazon Turk with three separate tasks. For each task, we ask each user 24 questions. Each question is a forced single-choice selection with two options among our and one baseline result with the rendered images from the same 4 sampled views and is evaluated by 20 different users. We only show input photos for the first similarity question, and hide the input photos for the other two questions to make the user focus on texture quality. We summarize the results in Table 4.1. Our results show significant preference by the users in terms of image fidelity, texture photorealism, and shape-texture consistency.

We also evaluate the similarity via image-based CLIP feature [148] between reference and the rendered images of synthesized textures. The CLIP similarity has been applied to material matching [251] and stylization [142]. A good transfer should transfer only the texture from images and should take into account the target shape geometry and transfer the texture semantically. For example, the transfer should be painted with respect to each part of the shape. We use our evaluation set to compute the comparison. For each image set and target 3D mesh pair, we compute the average of the metric among each reference image and each of rendered image from 4 sampled views (*i.e.* left front, right front, left back, and right back). We average the CLIP similarity across all (image set, mesh) pairs. Table 4.2 shows our method has the highest CLIP similarity.


**Figure 4.8.** Diversity of synthesized textures.

### Ablation study

We qualitatively perform an ablation study in Figure 4.7. The results suffer from geometry-texture misalignment without ControlNet or the depth-based ControlNet. Only normal-based ControlNet can accurately control the synthesized texture to be consistent with the input mesh geometry. We validate the importance of score distillation sampling. Only using SDS in our framework cannot achieve enough fidelity and the result tends to be blurry. Without LoRA removed (which is usually optimized with vanilla VSD), the optimization tends to make the distribution diverge from the personalized diffusion model. It makes the output contain less original texture but more irrelevant patterns from the input. We hypothesize that optimizing LoRA weights with a text condition containing a rare identifier tends to drive the distribution of rendered images to have a rare appearance.

We analyze the effectiveness of ControlNet and the design of score distillation in Section 4.4.3. We perform an additional ablation study in Figure 4.10. If we replace generic diffusion model $\epsilon_\phi$ with the personalized diffusion model $\epsilon_\psi$ or apply classifier free guidance weight 7.5, the result tends to introduce random patterns which does not exist in the input images. If we choose to freeze the camera encoder weights $\rho$, the result becomes worse or more noisy than our full method.

We also quantitatively evaluate the importance of each component in our system, as

**Figure 4.9.** Captured shapes. Input images are from Figure 4.6.

**Table 4.3.** Ablation study on image-based texturing w.r.t. CLIP image-based feature similarity. Although *w/o ControlNet* and *w/ ControlNet (Depth)* achieve higher similarity score, the transfer results tend to ignore target shape and directly paint the texture without reasoning the geometry. Among the remaining ablative methods, our full method achieves the highest CLIP similarity w.r.t. reference images.

| | CLIP similarity ↑ |
|---|---|
| w/o ControlNet | *0.8394* |
| w/ ControlNet (Depth) | *0.8320* |
| SDS, w/o CFG | 0.8101 |
| SDS, CFG 100 | 0.7983 |
| w/o LoRA removed | 0.8110 |
| Personalized model as $\phi$ | 0.8218 |
| CFG weight as 7.5 | 0.8218 |
| w/o camera encoder $\rho$ updated | 0.8267 |
| Ours | **0.8296** |

shown in Table 4.3. We use image-based CLIP feature to measure the similarity between reference images and the rendered images. To ensure fair evaluation, the background of both reference and rendered images are masked with white color. Our full method achieves the highest similarity score among the ablative baselines except *w/o ControlNet* and *w/ ControlNet (Depth)*. As shown in Figure 4.7, these two methods tend to ignore the target shape and directly paint the texture without adapting to geometry. Thus, they could reach higher score by painting the original texture regardless of the shape. We also observe that SDS results tend to be saturated or blurry and cannot recover the texture from the inputs. Keeping LoRA in the generic diffusion

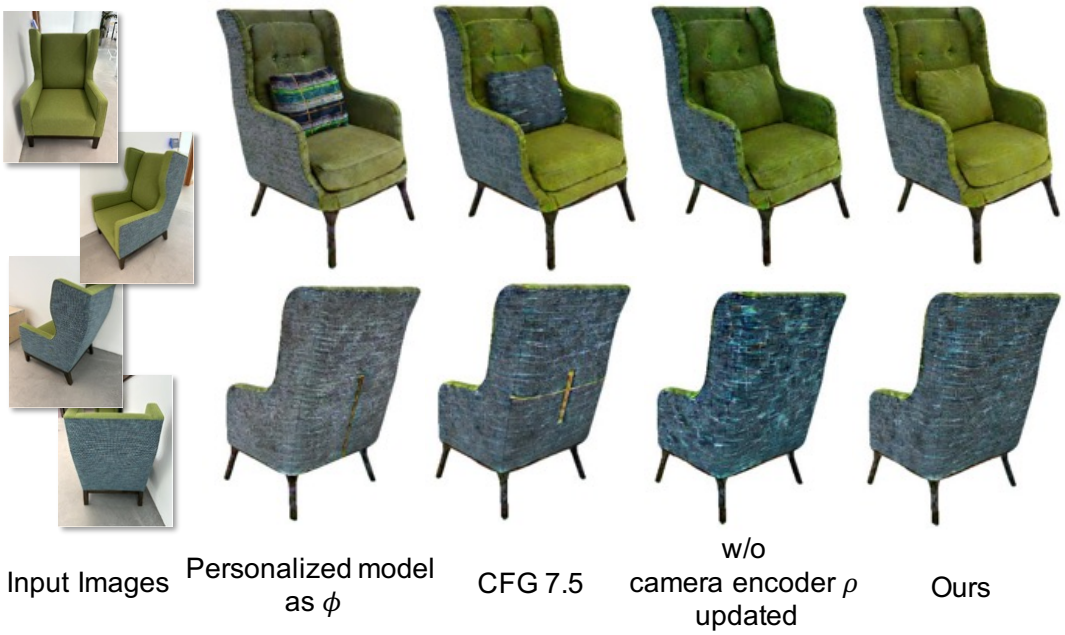



**Figure 4.10.** Additional Ablation study. If replacing generic diffusion model $\phi$ with personalized model or applying classifier guidance scale 7.5, some random patterns might appear in the synthesized texture. If we freeze the camera encoder $\rho$, the result might be worse or more noisy than our full method.


**Figure 4.11.** Number of images. Input images are from Figure 4.6.

model $\varepsilon_\phi$ will introduce random patterns to the synthesized texture.

### Number of images

We evaluate various number of input images. In Figure 4.11, a single image cannot provide sufficient information for texturing, while using 11 images doesn't show advantages in terms of texture quality.


**Figure 4.12.** Limitations. Our method may bake-in lighting into texture, have Janus problems when lacking enough input viewpoints, and ignore special and non-repeated patterns from the input.

## 4.5 Discussions

We proposed a framework to transfer high-quality texture from input images to an arbitrary shape. There are still some limitations, as shown in Figure 4.12. Our method may not be able to transfer special and non-repeated texture to the target shapes. Our method tends to bake in lighting to texture when there are strong specular highlights in the input images. Janus problem might appear when the viewpoints of input images do not cover the entire object. Nevertheless, we believe that our method can be the first step to tackling this challenging problem and will make an impact on the 3D content creation community.

Chapter 4 is based on the material as it appears in arXiv preprint arXiv:2401.09416. (”TextureDreamer: Image-guided Texture Synthesis through Geometry-aware Diffusion”, Yu-Ying Yeh, Jia-Bin Huang, Changil Kim, Lei Xiao, Thu Nguyen-Phuoc, Numair Khan, Cheng Zhang, Manmohan Chandraker, Carl S Marshall, Zhao Dong, Zhengqin Li). The dissertation author was the primary investigator and author of this paper.

# Chapter 5

# Neural 3D Reconstruction of Transparent Shapes

## 5.1 Introduction

Transparent objects abound in real-world environments, thus, their reconstruction from images has several applications such as 3D modeling and augmented reality. However, their visual appearance is far more complex than that of opaque objects, due to complex light paths with both refractions and reflections. This makes image-based reconstruction of transparent objects extremely ill-posed, since only highly convoluted intensities of an environment map are observed. In this chapter, we propose that data-driven priors learned by a deep network that models the physical basis of image formation can solve the problem of transparent shape reconstruction using a few natural images acquired with a commodity mobile phone camera.

While physically-based networks have been proposed to solve inverse problems for opaque objects [124], the complexity of light paths is higher for transparent shapes and small changes in shape can manifest as severely non-local changes in appearance. However, the physical basis of image formation for transparent objects is well-known – refraction at the interface is governed by Snell's law, the relative fraction of reflection is determined by Fresnel's equations and total internal reflection occurs when the angle of incidence at the interface to a medium with lower refractive index is below critical angle. These properties have been used to delineate theoretical conditions on reconstruction of transparent shapes [111], as well as acquire

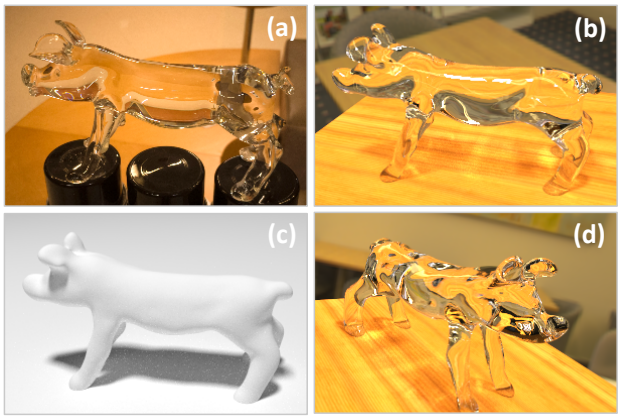



**Figure 5.1.** We present a novel physically-based deep network for image-based reconstruction of transparent objects with a small number of views. (a) An input photograph of a real transparent object captured under unconstrained conditions (1 of 10 images). (b) and (c): The reconstructed shape rendered under the same view with transparent and white diffuse material. (d) The reconstructed shape rendered under a novel view and environment map.


**Figure 5.2.** Reconstruction using 10 images of synthetic *kitten* model. The left image is rendered with the reconstructed shape while the right image is rendered with the ground-truth shape.

high-quality shapes under controlled settings [243, 258]. In contrast, we propose to leverage this knowledge of image formation within a deep network to reconstruct transparent shapes using relatively unconstrained images under arbitrary environment maps.

Specifically, we use a small number of views of a glass object with known refractive index, observed under a known but arbitrary environment map, using a mobile phone camera. Note that this is a significantly less restricted setting compared to most prior works that require dark room environments, projector-camera setups or controlled acquisition of a large number of images. Starting with a visual hull construction, we propose a novel in-network differentiable rendering layer that models refractive light paths up to two bounces to refine surface normals corresponding to a backprojected ray at both the front and back of the object, along with a mask to identify regions where total internal reflection occurs. Next, we propose a novel cost volume

to further leverage correspondence between the input image and environment map, but with special considerations since the two sets of normal maps span a four-dimensional space, which makes conventional cost volumes from multiview stereo intractable. Using our differentiable rendering layer, we perform a novel optimization in latent space to regularize our reconstructed normals to be consistent with the manifold of natural shapes. To reconstruct the full 3D shape, we use PointNet++ [172] with novel mechanisms to map normal features to a consistent 3D space, new loss functions for training and architectural changes that exploit surface normals for better recovery of 3D shape.

Since acquisition of transparent shapes is a laborious process, it is extremely difficult to obtain large-scale training data with ground truth [213]. Thus, we render a synthetic dataset, using a custom GPU-accelerated ray tracer. To avoid category-specific priors, we render images of random shapes under a wide variety of natural environment maps. On both synthetic and real data, the benefits of our physically-based network design are clearly observed. Indeed, we posit that such physical modeling eases the learning for a challenging problem and improves generalization to real images. Figures 5.1 and 5.2 show example outputs on real and synthetic data. All code and data will be publicly released.

To summarize, we propose the following contributions that solve the problem of transparent shape reconstruction with a limited number of unconstrained views:

- A physically-based network for surface normal reconstruction with a novel differentiable rendering layer and cost volume that imbibe insights from image formation.
- A physically-based 3D point cloud reconstruction that leverages the above surface normals and rendering layer.
- Strong experimental demonstration using a photorealistically rendered large-scale dataset for training and a small number of mobile phone photographs for evaluation.

## 5.2 Related Work

**Multiview stereo**

Traditional approaches [193] and deep networks [253] for multiview stereo have achieved impressive results. A full review is out of our scope, but we note that they assume photoconsistency for opaque objects and cannot handle complex light paths of transparent shapes.

**Theoretical studies**

In seminal work, Kutulakos and Steger [111] characterize the extent to which shape may be recovered given the number of bounces in refractive (and specular) light paths. Chari and Sturm [25] further constrain the system of equations using radiometric cues. Other works study motion cues [17, 150] or parametric priors [225]. We derive inspiration from such works to incorporate physical properties of image formation, by accounting for refractions, reflections and total internal reflections in our network design.

**Controlled acquisition**

Special setups have been used in prior work, such as light field probes [237], polarimetry [37, 86, 147], transmission imaging [105], scatter-trace photography [151], time-of-flight imaging [219] or tomography [224]. An external liquid medium [69] or moving spotlights in video [256] have been used too. Wu et al. [243] also start from a visual hull like us, to estimate normals and depths from multiple views acquired using a turntable-based setup with two cameras that image projected stripe patterns in a controlled environment. A projector-camera setup is also used by [176]. In contrast to all of the above works, we only require unconstrained natural images, even obtainable with a mobile phone camera, to reconstruct transparent shapes.

**Environment matting**

Environment matting uses a projector-camera setup to capture a composable map [277, 36]. Subsequent works have extended to mutliple cameras [137], natural images [238], frequency [275] or wavelet domains [167], with user-assistance [258], compressive sensing to reduce the number of images [47, 174] or deep network to predict the refractive flow from a single image [28]. In contrast, we use a small number of unconstrained images acquired with a mobile phone

in arbitrary scenes, to produce full 3D shape.

**Reconstruction from natural images**

Stets et al. [212] propose a black-box network to reconstruct depth and normals from a single image. Shan et al. [198] recover height fields in controlled settings, while Yeung et al. [257] have user inputs to recover normals. In contrast, we recover high-quality full 3D shapes and normals using only a few images of transparent objects, by modeling the physical basis of image formation in a deep network.

**Refractive materials besides glass**

Polarization [31], differentiable rendering [26] and neural volumes [131] have been used for translucent objects, while specular objects have been considered under similar frameworks as transparent ones [87, 278]. Gas flows [9, 92], flames [88, 246] and fluids [64, 175, 265] have been recovered, often in controlled setups. Our experiments are focused on glass, but similar ideas might be applicable for other refractive media too.

## 5.3 Method

**Setup and assumptions**

Our inputs are $V$ images $\{I_v\}_{v=1}^{V}$ of a transparent object with known refractive index (IoR), along with segmentation masks $\{M_v\}_{v=1}^{V}$. We assume a known and distant, but otherwise arbitrary, environment map $E$. The output is a point cloud reconstruction $\mathscr{P}$ of the transparent shape. Note that our model is different from (3-2-2) triangulation [109] that requires two reference points on each ray for reconstruction, leading to a significant relaxation over prior works [243, 258] that need active lighting, carefully calibrated devices and controlled environments. We tackle this severely ill-posed problem through a novel physically-based network that models the image formation in transparent objects over three sub-tasks: shape initialization, cost volume for normal estimation and shape reconstruction.

To simplify the problem and due to GPU memory limits, we consider light paths with only up to two bounces, that is, either the light ray gets reflected by the object once before hitting

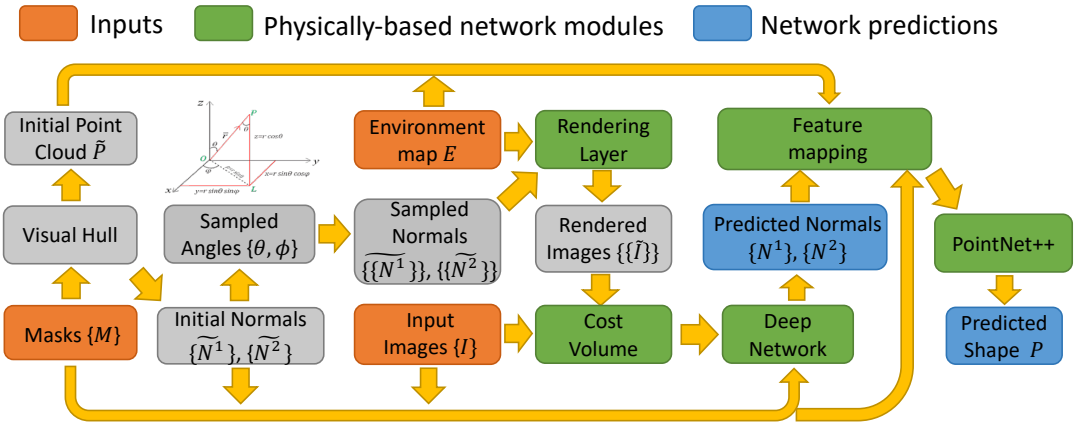



**Figure 5.3.** Our framework for transparent shape reconstruction.

the environment map or it gets refracted by it twice before hitting the environment map. This is not a severe limitation – more complex regions stemming from total internal reflection or light paths with more than two bounces are masked out in one view, but potentially estimated in other views. The overall framework is summarized in Figure 5.3.

**Shape initialization**

We initialize the transparent shape with a visual hull [110]. While a visual hull method cannot reconstruct some concave or self-occluded regions, it suffices as initialization for our network. We build a 3D volume of size $128^3$ and project segmentation masks from $V$ views to it. Then we use marching cubes to reconstruct the hull and loop L3 subdivision to obtain smooth surfaces.

### 5.3.1 Normal Reconstruction

A visual hull reconstruction from limited views might be inaccurate, besides missed concavities. We propose to reconstruct high quality normals by estimating correspondences between the input image and the environment map. This is a very difficult problem, since different configurations of transparent shapes may lead to the same appearance. Moreover, small perturbations of normal directions can cause pixel intensities to be completely different. Thus,

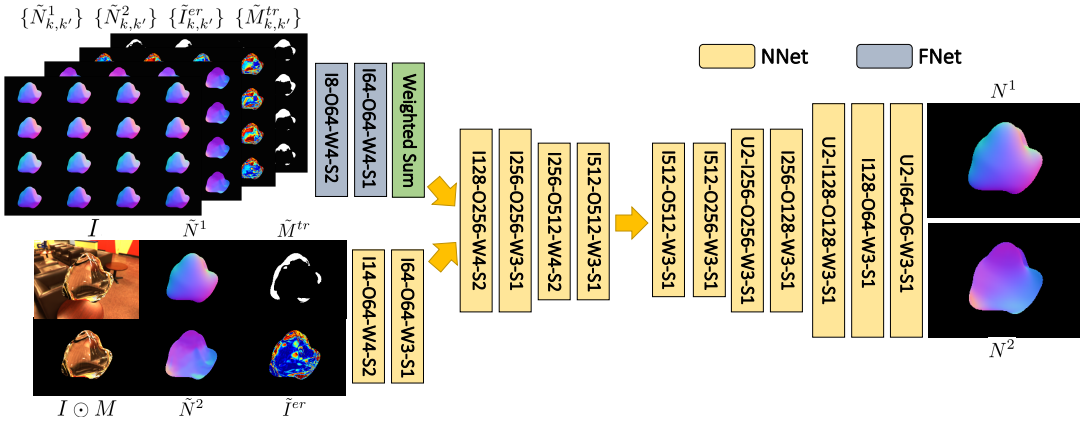



**Figure 5.4.** The network architecture for normal reconstruction. Yellow blocks represent **NNet** and blue blocks represent **FNet**. $IX_1$-$OX_2$-$WX_3$-$SX_4$ represents a convolutional layer with input channel $X_1$, output channel $X_2$, kernel size $X_3$ and stride $X_4$. $UX_5$ represents bilinear upsampling layer with scale factor $X_5$.

strong shape priors are necessary for a high quality reconstruction, which we propose to learn with a physically-inspired deep network.

**Basic network**

Our basic network architecture for normal estimation is shown in Figure 5.4. The basic network structure consists of one encoder and one decoder. The outputs of our network are two normal maps $N^1$ and $N^2$, which are the normals at the first and second hit points $P^1$ and $P^2$ for a ray backprojected from camera passing through the transparent shape, as illustrated in Figure 5.5(a). The benefit of modeling the estimation through $N^1$ and $N^2$ is that we can easily use a network to represent complex light transport effects without resorting to ray-tracing, which is time-consuming and difficult to treat differentiably. In other words, given $N^1$ and $N^2$, we can directly compute outgoing ray directions after passage through the transparent object. The inputs to our network are the image $I$, the image with background masked out $I \odot M$ and the $\tilde{N}^1$ and $\tilde{N}^2$ of the visual hull (computed off-line by ray tracing). We also compute $\hat{N}^1$ and $\hat{N}^2$ of the ground-truth shape for supervision. The definition of $\tilde{N}^1$, $\tilde{N}^2$ and $\hat{N}^1$, $\hat{N}^2$ are visualized in Figure 5.5(b). The basic network estimates:

$$N^1, N^2 = \textbf{NNet}(I, I \odot M, \tilde{N}^1, \tilde{N}^2) \tag{5.1}$$

The loss function is simply the $L_2$ loss for $N^1$ and $N^2$.

$$\mathscr{L}_N = ||N^1 - \hat{N}^1||_2^2 + ||N^2 - \hat{N}^2||_2^2 \tag{5.2}$$

**Rendering layer**

Given the environment map $E$, we can easily compute the incoming radiance through direction $l$ using bilinear sampling. This allows us to build a differentiable rendering layer to model the image formation process of refraction and reflection through simple local computation. As illustrated in Figure 5.5(a), for every pixel in the image, the incident ray direction $l^i$ through that pixel can be obtained by camera calibration. The reflected and refracted rays $l^r$ and $l^t$ can be computed using $N^1$ and $N^2$, following Snell's law. Our rendering layer implements the full physics of an intersection, including the intensity changes caused by the Fresnel term $\mathscr{F}$ of the refractive material. More formally, with some abuse of notation, let $L^i$, $L^r$ and $L^t$ be the radiance of incoming, reflected and refracted rays. We have

$$\mathscr{F} = \frac{1}{2}\left(\frac{l^i \cdot N - \eta l^t \cdot N}{l^i \cdot N + \eta l^t \cdot N}\right)^2 + \frac{1}{2}\left(\frac{\eta l^i \cdot N - l^t \cdot N}{\eta l^i \cdot N + l^t \cdot N}\right)^2.$$
$$L^r = \mathscr{F} \cdot L^i, \qquad L^t = (1 - \mathscr{F}) \cdot L^i$$

Due to total internal reflection, some rays entering the object may not be able to hit the environment map after one more bounce, for which our rendering layer returns a binary mask, $M^{tr}$. With $I^r$ and $I^t$ representing radiance along the directions $l^r$ and $l^t$, the rendering layer models the image formation process for transparent shapes through reflection, refraction and total internal reflection:

$$I^r, I^t, M^{tr} = \textbf{RenderLayer}(E, N^1, N^2). \tag{5.3}$$

Our in-network rendering layer is differentiable and end-to-end trainable. But instead of just using the rendering loss as an extra supervision, we compute an error map based on rendering

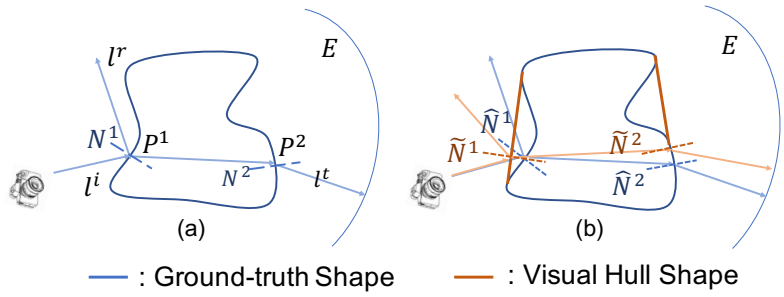



**Figure 5.5.** (a) Illustration of the first and second normal ($N^1$ and $N^2$), the first and second hit points ($P^1$ and $P^2$), and the reflection and refraction modeled by our deep network. (b) Illustration of visual hull ($\tilde{N}^1$, $\tilde{N}^2$) and ground-truth normals ($\hat{N}^1$, $\hat{N}^2$).

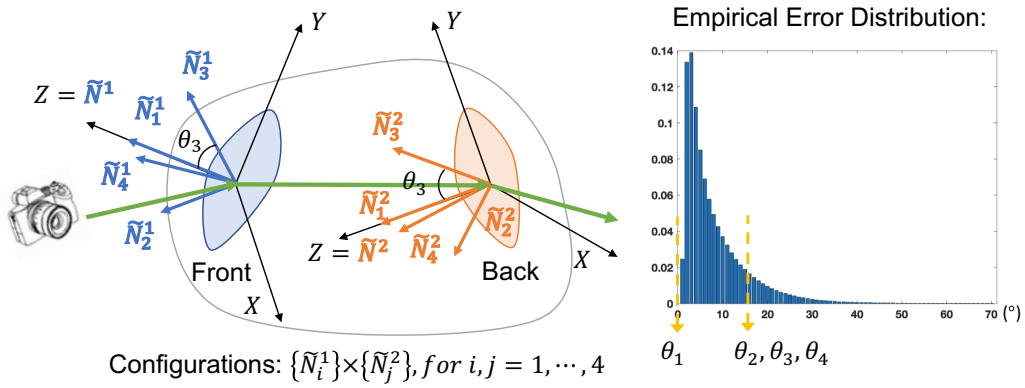



**Figure 5.6.** We build an efficient cost volume by sampling directions around visual hull normals according to their error distributions.

with the visual hull normals:

$$\tilde{I}^r, \tilde{I}^t, \tilde{M}^{tr} = \textbf{RenderLayer}(E, \tilde{N}^1, \tilde{N}^2), \tag{5.4}$$

$$\tilde{I}^{er} = |I - (\tilde{I}^r + \tilde{I}^t)| \odot M. \tag{5.5}$$

This error map is used as an additional input to our normal reconstruction network, to help it better learn regions where the visual hull normals $\tilde{N}^1$ and $\tilde{N}^2$ may not be accurate:

$$N^1, N^2 = \textbf{NNet}(I, I \odot M, \tilde{N}^1, \tilde{N}^2, \tilde{I}^{er}, \tilde{M}^{tr}) \tag{5.6}$$

**Cost volume**

We now propose a cost volume to leverage the correspondence between the environment map and the input image. While cost volumes in deep networks have led to great success for multiview depth reconstruction of opaque objects, extension to normal reconstruction for transparent objects is non-trivial. The brute-force approach would be to uniformly sample the 4-dimensional hemisphere of $N^1 \times N^2$, then compute the error map for each sampled normal. However, this will lead to much higher GPU memory consumption compared to depth reconstruction due to higher dimensionality of the sampled space. To limit memory consumption, we sample $N^1$ and $N^2$ in smaller regions around the initial visual hull normals $\tilde{N}^1$ and $\tilde{N}^2$, as shown in Figure 5.6. Formally, let $U$ be the up vector in bottom-to-top direction of the image plane. We first build a local coordinate system with respect to $\tilde{N}^1$ and $\tilde{N}^2$:

$$Z = \tilde{N}^i,\ Y = U - (U^T \cdot \tilde{N}^i)\tilde{N}^i,\ X = \texttt{cross}(Y,Z), \tag{5.7}$$

where $Y$ is normalized and $i = 1,2$. Let $\{\theta_k\}_{k=1}^K$, $\{\phi_k\}_{k=1}^K$ be the sampled angles. Then, the sampled normals are:

$$\tilde{N}_k^i = X\cos\phi_k \sin\theta_k + Y \sin\phi_k \sin\theta_k + Z\cos\theta_k. \tag{5.8}$$

We sample the angles $\{\theta_k\}_{k=1}^K$, $\{\phi_k\}_{k=1}^K$ according to the error distribution of visual hull normals. The angles and distributions are shown in the supplementary material. Since we reconstruct $N^1$ and $N^2$ simultaneously, the total number of configurations of sampled normals is $K \times K$. Directly using the $K^2$ sampled normals to build a cost volume is too expensive, so we use a learnable pooling layer to aggregate the features from each sampled normal configuration in an early stage. For each pair of $\tilde{N}_k^1$ and $\tilde{N}_{k'}^2$, we compute their total reflection mask $\tilde{M}_{k,k'}^{tr}$ and error map $\tilde{I}_{k,k'}^{er}$

using (5.4) and (5.5), then perform a feature extraction:

$$F(k,k') = \mathbf{FNet}(\tilde{N}^1_k, \tilde{N}^2_{k'}, \tilde{I}^{er}_{k,k'}, \tilde{M}^{tr}_{k,k'}). \tag{5.9}$$

We then compute the weighted sum of feature vectors $F(k,k')$ and concatenate them with the feature extracted from the encoder of **NNet** for normal reconstruction:

$$F = \sum_{k}^{K}\sum_{k'}^{K} \omega(k,k')F(k,k'), \tag{5.10}$$

where $\omega(k,k')$ are positive coefficients with sum equal to 1, that are also learned during the training process. The detailed network structure is shown in Figure 5.4.

**Post processing**

The network above already yields reasonable normal reconstruction. It can be further improved by optimizing the latent vector from the encoder to minimize the rendering error using the predicted normal $N^1$ and $N^2$:

$$\mathscr{L}_N^{\text{Opt}} = ||(I - (I^r + I^t)) \odot M^{tr}||_2^2, \tag{5.11}$$

where $I_t, I_t, M^{tr}$ are obtained from the rendering layer (5.3). For this optimization, we keep the network parameters unchanged and only update the latent vector. Note that directly optimizing the predicted normal $N^1$ and $N^2$ without the deep network does not yield comparable improvements. This is due to our decoder acting as a regularization that prevents the reconstructed normal from deviating from the manifold of natural shapes during the optimization. Similar ideas have been used for BRDF reconstruction [56].

### 5.3.2 Point Cloud Reconstruction

We now reconstruct the transparent shape based on the predictions of **NNet**, that is, the normals, total reflection mask and rendering error. Our idea is to map the predictions from

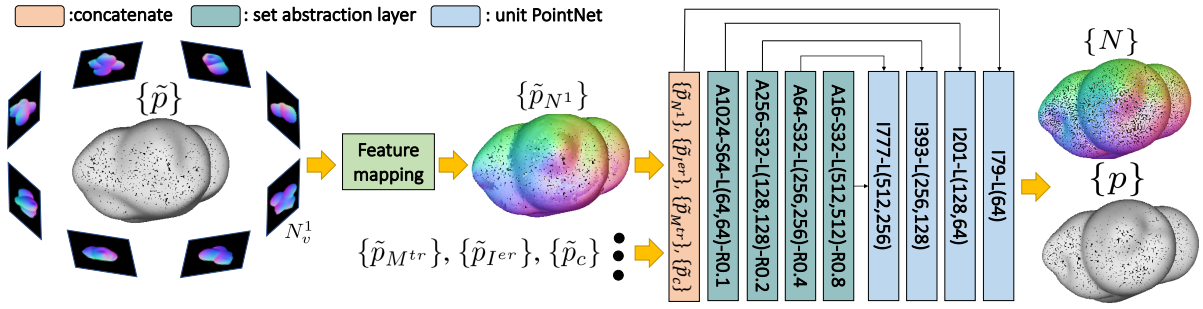



**Figure 5.7.** Our method for point cloud reconstruction. $AX_1$-$SX_2$-$L(X_3, X_4)$-$RX_5$ represents a set abstraction layer with $X_1$ anchor points, $X_2$ sampled points, 2 fully connected layers with $X_3$, $X_4$ feature channels and sampling radius $X_5$. $IY_1$-$L(Y_2, Y_3)$ represents a unit PointNet with $Y_1$ input channels and 2 fully connected layers with $Y_2$, $Y_3$ feature channels.

different views to the visual hull geometry. These predictions are used as input features for a point cloud reconstruction to obtain a full 3D shape. Our point cloud reconstruction pipeline is illustrated in Figure 5.7.

**Feature mapping**

We propose three options to map predictions from different views to the visual hull geometry. Let $\{\tilde{p}\}$ be the point cloud uniformly sampled from visual hull surfaces and $\mathscr{S}_v(\tilde{p}, h)$ be a function that projects the 3D point $\tilde{p}$ to the 2D image plane of view $v$ and then fetches the value of a function $h$ defined on image coordinates using bilinear sampling. Let $\mathscr{V}_v(\tilde{p})$ be a binary function that verifies if point $\tilde{p}$ can be observed from view $v$ and $\mathscr{T}_v(\tilde{p})$ be a transformation that maps a 3D point or normal direction in view $v$ to world coordinates. Let $\mathscr{C}_v(\tilde{p})$ be the cosine of the angle between the ray passing through $\tilde{p}$ and camera center.

The first option is a feature $f$ that averages observations from different views. For every view $v$ that can see the point $\tilde{p}$, we project its features to the point and compute a mean:

$$\tilde{p}_{N^1} = \frac{\sum_v \mathscr{T}_v(\mathscr{S}_v(\tilde{p}, N_v^1))\mathscr{V}_v(\tilde{p})}{\sum_v \mathscr{V}_v(\tilde{p})}, \quad \tilde{p}_{I^{er}} = \frac{\sum_v \mathscr{S}_v(\tilde{p}, I_v^{er})\mathscr{V}_v(\tilde{p})}{\sum_v \mathscr{V}_v(\tilde{p})},$$
$$\tilde{p}_{M^{tr}} = \frac{\sum_v \mathscr{S}_v(\tilde{p}, M_v^{tr})\mathscr{V}_v(\tilde{p})}{\sum_v \mathscr{V}_v(\tilde{p})}, \quad \tilde{p}_c = \frac{\sum_v \mathscr{C}_v(\tilde{p})\mathscr{V}_v(\tilde{p})}{\sum_v \mathscr{V}_v(\tilde{p})}.$$

We concatenate to get: $f = [\tilde{p}_{N^1}, \tilde{p}_{I^{er}}, \tilde{p}_{M^{tr}}, \tilde{p}_c]$.

Another option is to select a view $v^*$ with potentially the most accurate predictions and compute $f$ using the features from only that view. We consider two view-selection strategies. The first is nearest view selection, in which we simply select $v^*$ with the largest $\mathscr{C}_v(\tilde{p})$. The other is to choose the view with the lowest rendering error and no total reflection, with the algorithm detailed in supplementary material. Note that although we do not directly map $N^2$ to the visual hull geometry, it is necessary for computing the rendering error and thus, needed for our shape reconstruction.

**Point cloud refinement**

We build a network following PointNet++ [172] to reconstruct the point cloud of the transparent object. The input to the network is the visual hull point cloud $\{\tilde{p}\}$ and the feature vectors $\{f\}$. The outputs are the normals $\{N\}$ and the offset of visual hull points $\{\delta\tilde{p}\}$, with the final vertex position is computed as $p = \tilde{p} + \delta\tilde{p}$:

$$\{\delta\tilde{p}\}, \{N\} = \textbf{PNet}(\{\tilde{p}\}, \{f\}). \tag{5.12}$$

We tried three loss functions to train our PointNet++. The first loss function is the nearest $L_2$ loss $\mathscr{L}_P^{\text{nearest}}$. Let $\hat{p}$ be the nearest point to $\tilde{p}$ on the surface of ground-truth geometry and $\hat{N}$ be its normal. We compute the weighted sum of $L_2$ distance between our predictions $p$, $N$ and ground truth:

$$\mathscr{L}_P^{\text{nearest}} = \sum_{\{p\},\{N\}} \lambda_1 ||p - \hat{p}||_2^2 + \lambda_2 ||N - \hat{N}||_2^2. \tag{5.13}$$

The second loss function is a view-dependent $L_2$ loss $\mathscr{L}_P^{\text{view}}$. Instead of choosing the nearest point from ground-truth geometry for supervision, we choose the point from the best view $v^*$ by projecting its geometry into world coordinates:

$$\hat{p}_{v^*}, \hat{N}_{v^*} = \begin{cases} \mathscr{T}_{v^*}(\mathscr{S}_{v^*}(\tilde{p}, \hat{P}^1_{v^*})), \mathscr{T}_{v^*}(\mathscr{S}_{v^*}(\tilde{p}, \hat{N}^1_{v^*})), & v^* \neq 0 \\ \hat{p}, \hat{N}, & v^* = 0. \end{cases}$$

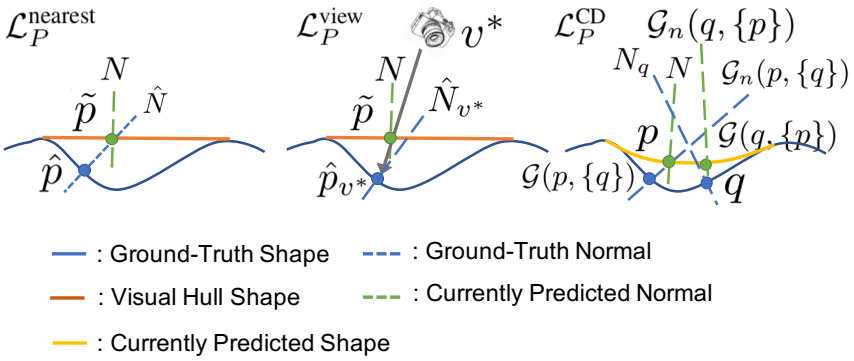



**Figure 5.8.** A visualization of the loss functions for point cloud reconstruction. From left to the right are nearest $L_2$ loss $\mathscr{L}_P^{\text{nearest}}$, view-dependent $L_2$ loss $\mathscr{L}_P^{view}$ and chamfer distance loss $\mathscr{L}_P^{\text{CD}}$.

Then we have

$$\mathscr{L}_P^{\text{view}} = \sum_{\{p\},\{N\}} \lambda_1 ||p - \hat{p}_{v^*}||_2^2 + \lambda_2 ||N - \hat{N}_{v^*}||_2^2. \tag{5.14}$$

The intuition is that since both the feature and ground-truth geometry are selected from the same view, the network can potentially learn their correlation more easily. The last loss function, $\mathscr{L}_P^{\text{CD}}$, is based on the Chamfer distance. Let $\{q\}$ be the set of points uniformly sampled from the ground-truth geometry, with normals $N_q$. Let $\mathscr{G}(p,\{q\})$ be a function which finds the nearest point of $p$ in the point set $\{q\}$ and function $\mathscr{G}_n(p,\{q\})$ return the normal of the nearest point. The Chamfer distance loss is defined as

$$\begin{aligned}\mathscr{L}_P^{\text{CD}} = &\sum_{\{p\},\{N\}} \frac{\lambda_1}{2} ||p - \mathscr{G}(p,\{q\})|| + \frac{\lambda_2}{2} ||N - \mathscr{G}_n(p,\{q\})|| + \\ &\sum_{\{q\},\{N_q\}} \frac{\lambda_1}{2} ||q - \mathscr{G}(q,\{p\})|| + \frac{\lambda_2}{2} ||N_q - \mathscr{G}_n(q,\{p\})||.\end{aligned} \tag{5.15}$$

Figure 5.8 is a demonstration of the three loss functions. In all our experiments, we set $\lambda_1 = 200$ and $\lambda_2 = 5$.

Our network, shown in Figure 5.7, makes several improvements over standard PointNet++. First, we replace max-pooling with average-pooling to favor smooth results. Second, we concatenate normals $\{N\}$ to all skip connections to learn details. Third, we augment the input feature of set abstraction layer with the difference of normal directions between the current and

center points. Section 5.4 and supplementary material show the impact of our design choices.

## 5.4 Experiments

**Dataset**

We procedurally generate random scenes following [124, 250] rather than use shape repositories [24], to let the model be category-independent. To remove inner structures caused by shape intersections and prevent false refractions, we render 75 depth maps and use PSR [102] to fuse them into a mesh, with L3 loop subdivision to smooth the surface. We implement a physically-based GPU renderer using NVIDIA OptiX [3]. With 1499 HDR environment maps of [58] for training and 424 for testing, we render 3000 random scenes for training and 600 for testing. The IoR of all shapes is set to 1.4723, to match our real objects. Our experiments also include sensitivity analysis to characterize the behavior of the network when the test-time IoR differs from this value.

**Implementation Details**

When building the cost volume for normal reconstruction, we set the number of sampled angles $K$ to be 4. Increasing the number of sampled angles will drastically increase the memory consumption and does not improve the normal accuracy. We sample $\phi$ uniformly from 0 to $2\pi$ and sample $\theta$ according to the visual hull normal error. To build the cost volume (`cv`) for normal prediction, we sample $\phi$ uniformly from 0 to $2\pi$ and sample $\theta$ according to the visual hull normal error. In particular, we first randomly sample 100 scenes from our synthetic dataset and compute the angles between visual hull normals and ground truth normals. We set one $\theta$ value to be 0 and the other to larger than 85% of angles between the visual hull normal $\tilde{N}^1$ and ground truth normal $\hat{N}^1$. The distribution of visual hull normal $\tilde{N}^1$ error for 5, 10 and 20 views are presented in Figure 5.9. Table 5.1 summarizes the configurations of $\{\theta\}$ and $\{\phi\}$ angles for different number of views.

We use Adam optimizer to train all our networks. The initial learning rate is set to be $10^{-4}$ and we halve the learning rate every 2 epochs. All networks are trained over 10 epochs.

**Table 5.1.** The sampled angles for building cost volume. We set the sampled angles according to the normal error of visual hull reconstructed by different number of views.

| | $\{\theta_k\}_{k=1}^4$ | $\{\phi_k\}_{k=1}^4$ |
|---|---|---|
| 5 views | 0°,25°,25°,25° | 0°,0°,120°,240° |
| 10 views | 0°,15°,15°,15° | 0°,0°,120°,240° |
| 20 views | 0°,10°,10°,10° | 0°,0°,120°,240° |

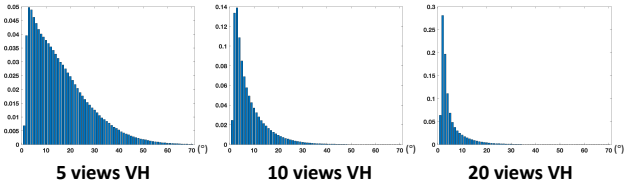



**Figure 5.9.** The error distribution of visual hull normals $\tilde{N}^1$ from different number of views.

### 5.4.1 Ablation Studies on Synthetic Data

#### Normal reconstruction

The quantitative comparisons of 10 views normal reconstruction are summarized in Table 5.2. We report 5 metrics: the median and mean angles of the first and the second normals ($N^1$, $N^2$), and the mean rendering error ($I^{er}$). We first compare the normal reconstruction of the basic encoder-decoder structure with (`wr`) and without rendering error and total reflection mask as input (`basic`). While both networks greatly improve the normal accuracy compared to visual hull normals (`vh10`), adding rendering error and total reflection mask as inputs can help achieve overall slightly better performances. Next we test the effectiveness of the cost volume (`wr+cv`). Quantitative numbers show that adding cost volume achieves better results, which coincides with our intuition that finding the correspondences between input image and the environment map can help our normal prediction. Finally we optimize the latent vector from the encoder by minimizing the rendering error (`wr+cv+op`). It significantly reduces the rendering error and also improves the normal accuracy. Such improvements cannot be achieved by directly optimizing the normal predictions $N^1$ and $N^2$ in the pixel space. Figure 5.10 presents normal reconstruction results from our synthetic dataset. Our normal reconstruction pipeline obtains results of much higher quality compared with visual hull method. Ablation studies of 5 views and 20 views normal reconstruction and the optimization of latent vector are included in Section 5.4.4 and 5.4.4.

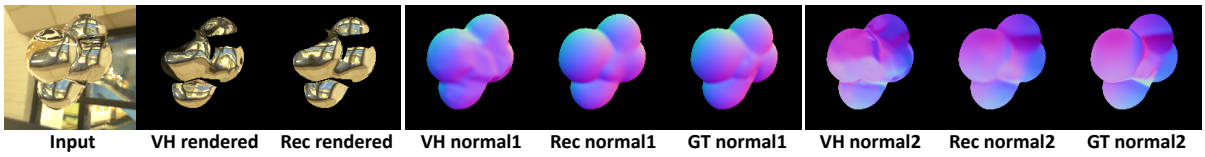



**Figure 5.10.** An example of 10 views normal reconstruction from our synthetic dataset. The region of total reflection has been masked out in the rendered images.

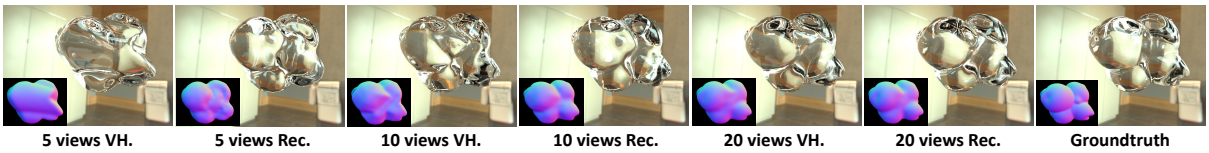



**Figure 5.11.** Our transparent shape reconstruction results from 5 views, 10 views and 20 views from our synthetic dataset. The images rendered with our reconstructed shapes are much closer to the ground-truth compared with images rendered with the visual hull shapes. The inset normals are rendered from the reconstructed shapes.

### Point cloud reconstruction

Quantitative comparisons of the 10-view point cloud reconstruction network are summarized in Table 5.3. After obtaining the point and normal predictions $\{p\}$ and $\{N\}$, we reconstruct 3D meshes as described above. We compute the Chamfer distance (CD), Chamfer normal median angle (CDN-med), Chamfer normal mean angle (CDN-mean) and Metro distance by uniformly sampling 20000 points on the ground-truth and reconstructed meshes. We first compare the effectiveness of different loss functions. We observe that while all the three loss functions can greatly improve the reconstruction accuracy compared with the initial 10-view visual hull, the Chamfer distance loss (RE-$\mathscr{L}_P^{\text{CD}}$) performs significantly better than view-dependent loss (RE-$\mathscr{L}_P^{\text{view}}$) and nearest $L_2$ loss (RE-$\mathscr{L}_P^{\text{nearest}}$). Next, we test different feature mapping strategies, where the rendering error-based view selection method (RE-$\mathscr{L}_P^{\text{CD}}$) performs consistently better than the other two methods. This is because our rendering error can be used as a meaningful metric to predict normal reconstruction accuracy, which leads to better point cloud reconstruction. Ablation studies for the modified PointNet++ are included in Section 5.4.4.

The last row of Table 5.3 shows that an optimization-based method like PSR [102] to refine shape from predicted normals does not lead to much improvement, possibly since visual

**Table 5.2.** Quantitative comparisons of normal estimation from 10 views. `vh10` represents the initial normals reconstructed from 10 views visual hull. `wr` and `basic` are our basic encoder-decoder network with and without rendering error map ($I^{er}$) and total reflection mask ($M^{tr}$) as inputs. `wr+cv` represents our network with cost volume. `wr+cv+op` represents the predictions after optimizing the latent vector to minimize the rendering error. `wr+cv var. IoR` represents sensitivity analysis for IoR, explained in text.

| | `vh10` | `basic` | `wr` | `wr+cv` | `wr+cv +op` | `wr+cv var. IoR` |
|---|---|---|---|---|---|---|
| $N^1$ median (°) | 5.5 | 3.5 | 3.5 | 3.4 | **3.4** | 3.6 |
| $N^1$ mean (°) | 7.5 | 4.9 | 5.0 | 4.8 | **4.7** | 5.0 |
| $N^2$ median (°) | 9.2 | 6.9 | 6.8 | 6.6 | **6.2** | 7.3 |
| $N^2$ mean (°) | 11.6 | 8.8 | 8.7 | 8.4 | **8.1** | 9.1 |
| Render Err.($10^{-2}$) | 6.0 | 4.7 | 4.6 | 4.4 | **2.9** | 5.5 |

**Table 5.3.** Quantitative comparisons of point cloud reconstruction from 10 views. `RE`, `NE` and `AV` represent feature mapping methods: rendering error based view selection, nearest view selection and average fusion, respectively. $\mathscr{L}_P^{\text{nearest}}$, $\mathscr{L}_P^{\text{view}}$ and $\mathscr{L}_P^{\text{CD}}$ are the loss functions defined in Sec. 5.3. `RE-`$\mathscr{L}_P^{\text{CD}}$, `var. IoR` represents sensitivity analysis for IoR, as described in text. PSR represents optimization [102] to refine the point cloud based on predicted normals.

| | CD($10^{-4}$) | CDN-mean(°) | CDN-med(°) | Metro($10^{-3}$) |
|---|---|---|---|---|
| `vh10` | 5.14 | 7.19 | 4.90 | 15.2 |
| `RE-`$\mathscr{L}_P^{\text{nearest}}$ | 2.17 | 6.23 | 4.50 | 7.07 |
| `RE-`$\mathscr{L}_P^{\text{view}}$ | 2.15 | 6.51 | 4.76 | 6.79 |
| `RE-`$\mathscr{L}_P^{\text{CD}}$ | **2.00** | **6.02** | **4.38** | **5.98** |
| `NE-`$\mathscr{L}_P^{\text{CD}}$ | 2.04 | 6.10 | 4.46 | 6.02 |
| `AV-`$\mathscr{L}_P^{\text{CD}}$ | 2.03 | 6.08 | 4.46 | 6.09 |
| `RE-`$\mathscr{L}_P^{\text{CD}}$, `var. IoR` | 2.13 | 6.24 | 4.56 | 6.11 |
| PSR | 5.13 | 6.94 | 4.75 | 14.7 |

hull shapes are still significantly far from ground truth. In contrast, our network allows large improvements.

**Different number of views**

We also test the entire reconstruction pipeline for 5 and 20 views, with results summarized in Table 5.4. We use the setting that leads to the best performance for 10 views, that is, `wr` + `cv` + `op` for normal reconstruction and `RE-`$\mathscr{L}_P^{\text{CD}}$ for point cloud reconstruction, achieving significantly lower errors than the visual hull method. Figure 5.11 shows an example from the synthetic test set for reconstructions with different number of views. Further results and

**Table 5.4.** Quantitative comparisons of point cloud reconstruction from 5 views and 20 views. In both cases, our pipeline significantly improves the transparent shape reconstruction accuracy compared with classical visual hull method.

| | CD($10^{-4}$) | CDN-mean(°) | CDN-med(°) | Metro($10^{-3}$) |
|---|---|---|---|---|
| vh5 | 31.7 | 13.1 | 10.3 | 66.6 |
| Rec5 | **6.30** | **11.0** | **8.7** | **15.2** |
| vh20 | 2.23 | 4.59 | **2.71** | 6.83 |
| Rec20 | **1.20** | **4.04** | 2.73 | **4.18** |

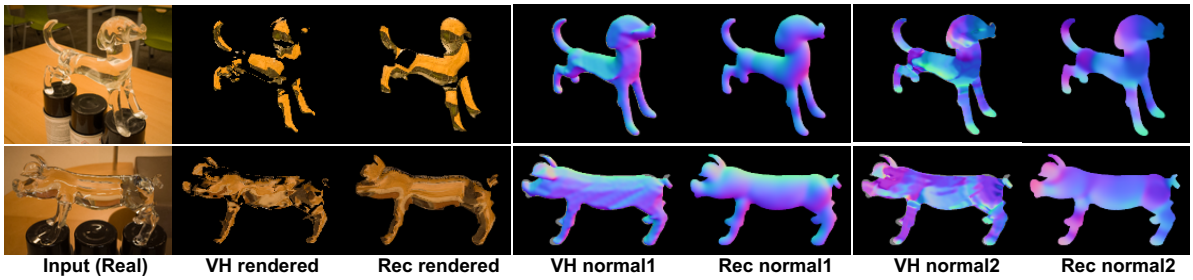



**Figure 5.12.** Normal reconstruction of real transparent objects and the rendered images. The initial visual hull normals are built from 10 views. The region of total reflection has been masked out in the rendered images.

comparisons are in Section 5.4.4.

**Sensitivity analysis for IoR**

We also evaluate the model on another test set with the same geometries, but unknown IoRs sampled uniformly from the range $[1.3, 1.7]$. As shown in Tables 5.2 and 5.3, errors increase slightly but stay reasonable, showing that our model can tolerate inaccurate IoRs to some extent. More details are provided in Section 5.4.5.

### 5.4.2 Results on Real Transparent Objects

We acquire RGB images using a mobile phone. To capture the environment map, we take several images of a mirror sphere at the same location as the transparent shape. We use COLMAP [191] to obtain the camera poses and manually create the segmentation masks.

**Normal reconstruction**

We first demonstrate the normal reconstruction results on real transparent objects in Figure 5.12. Our model significantly improves visual hull normal quality. The images rendered

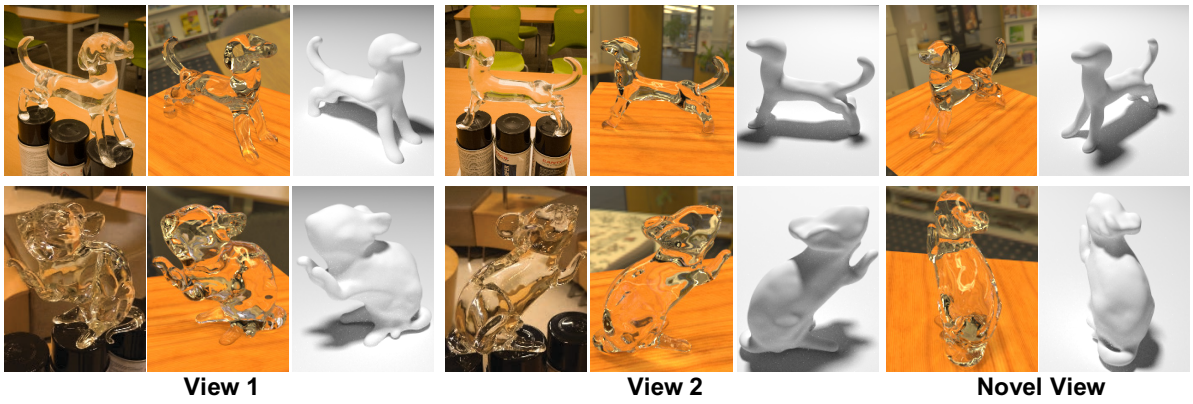



**Figure 5.13.** 3D shape reconstruction on real data. Columns 1-6 show reconstruction results from 2 known view directions. For each view, we show the input image and the reconstructed shape rendered from the same view under different lighting and materials. Columns 7-8 render the reconstructed shape from a novel view direction that has not been used to build the visual hull. The first shape is reconstructed using only 5 views (top row) while the second uses 10 views (bottom row). Also, see comparisons to ground truth scans in Section 5.4.3.

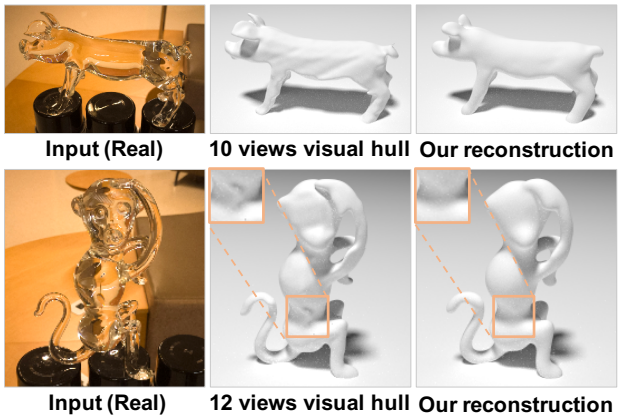



**Figure 5.14.** Comparison between visual hull initialization and our shape reconstruction on real objects. Our method recovers more details, especially for concave regions.

from our predicted normals are much more similar to the input RGB images compared to those rendered from visual hull normals.

**3D shape reconstruction**

In Figure 5.13, we demonstrate our 3D shape reconstruction results on real world transparent objects under natural environment map. The dog shape in the first row only takes 5 views and the mouse shape in the second row takes 10 views. We first demonstrate the reconstructed shape from the same view as the input images by rendering them under different lighting and materials. Even with very limited inputs, our reconstructed shapes are still of high quality. To test the generalizability of our predicted shapes, we render them from novel views that have not been used as inputs and the results are still reasonable. Figure 5.14 compares our reconstruction results with the visual hull initialization. We observe that our method performs much better, especially for concave regions. Comparisons with scanned ground truth are demonstrated in Section 5.4.3.

**Runtime**

Our method requires around 46s to reconstruct a transparent shape from 10 views on a 2080 Ti, compared to 5-6 hours for previous optimization-based methods [243].

### 5.4.3 Real Data Evaluation with Ground Truth

We first present quantitative comparisons and then more qualitative ones on real data. We use four objects (*mouse*, *dog*, *pig* and *monkey*). All objects are reconstructed from 10 views under natural environment maps, except *monkey*, which needs 12 views since the shape is much more complex. To obtain the ground truth geometry, we paint each object with diffuse white paint and scan using a high-quality 3D scanner. All code and data will be publicly released.

**Quantitative results**

We manually align ground-truth shapes with the predicted shapes using ICP method [18] and then uniformly sample 20000 points on the both shapes to compute the four error metrics (CD, CDN-mean, CDN-med, Metro). The quantitative numbers are summarized in Table 5.5.

**Table 5.5.** Quantitative comparisons of transparent shape reconstruction on real data. We observe that our reconstruction achieves lower average errors than the visual hull method on all the metrics.

| | Views | CD($10^{-4}$) | | CDN-mean(°) | | CDN-med(°) | | Metro($10^{-3}$) | |
|---|---|---|---|---|---|---|---|---|---|
| | | `vh` | `Rec` | `vh` | `Rec` | `vh` | `Rec` | `vh` | `Rec` |
| monkey | 12 | 3.99 | **3.94** | 21.2 | **16.4** | 14.8 | **11.9** | 20.7 | **13.9** |
| mouse | 10 | 8.04 | **5.35** | 19.0 | **16.3** | 11.4 | **12.0** | 16.6 | **13.0** |
| pig | 10 | 5.58 | **4.87** | 19.0 | **18.3** | **14.0** | 14.6 | 13.0 | **7.4** |
| dog | 10 | 2.25 | **1.86** | 14.5 | **12.4** | 11.4 | **10.3** | 4.1 | **4.0** |
| mean | 10.5 | 4.97 | **4.00** | 18.4 | **15.9** | 12.9 | **12.2** | 13.6 | **9.6** |

For all the 4 objects, our method consistently outperforms the visual hull baseline, which again demonstrates the effectiveness of our transparent shape reconstruction framework.

**Qualitative results and videos**

Figure 5.15 shows both the ground-truth transparent shapes and our reconstructed shapes rendered under different lighting and materials. Even though the shapes are complex and we use very limited inputs, our reconstructions still closely match the ground truth appearance. This demonstrates the efficacy of our physically-motivated network that models complex light paths induced by refractions and reflections. To better visualize the quality of our 3D reconstruction outputs, we create a video by rotating both the ground-truth shapes and the reconstructed shapes under different natural environment maps. The video is included in the supplementary material. A higher resolution video is available at this link.

### 5.4.4 Additional Ablation Studies

**Different number of views**

Table 5.6 summarizes the normal predictions from 5 and 20 views. Similar to the 10-view case, our entire method `wr+cv+op` outperforms all other baselines on all the five metrics. In particular, we find the cost volume (`cv`) and the optimization of the latent vector (`op`) bring the largest improvements on normal reconstruction accuracy. This justifies our intuition that utilizing the correspondence between the input image and the captured environment map by modeling


**Figure 5.15.** Results on 3D reconstruction for four real transparent objects. All shapes are reconstructed from 10 views, except the *monkey* in the last row that uses 12 views. We first present reconstruction results from two input views (columns 1-6). From left to right, the odd rows show the input image and the reconstructed shapes under different lighting and materials. The corresponding outputs using the ground-truth shapes rendered from the same view are shown in the even rows. We also render the reconstructed shapes and ground-truth shapes from a novel view direction that has not been used to build the visual hull (columns 7-8).

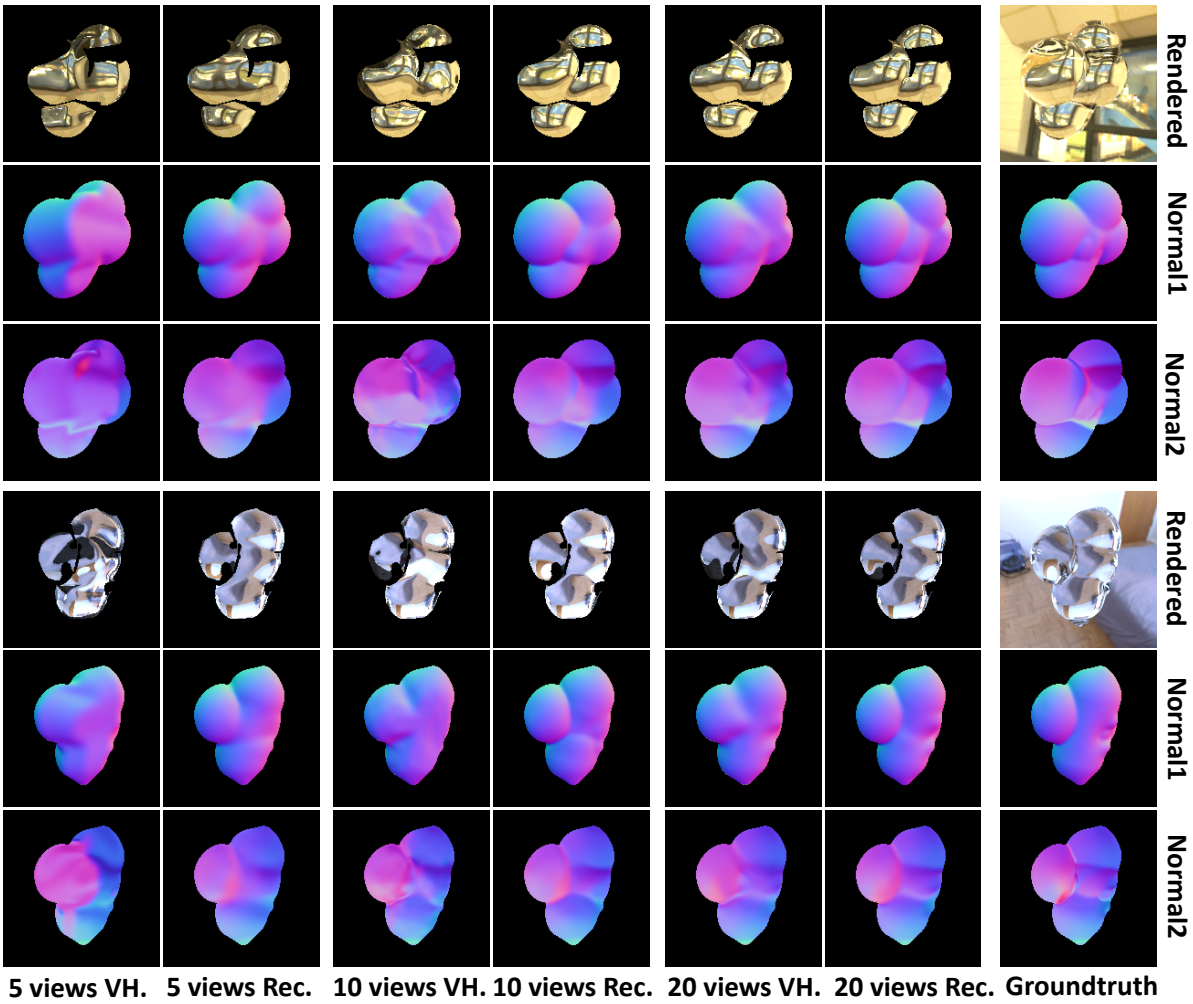



**Figure 5.16.** Normal predictions on our synthetic dataset with different number of input views. Regions with total reflection have been masked out in the rendered images. Our predicted normals are much closer to the ground truth compared to the visual hull normals.

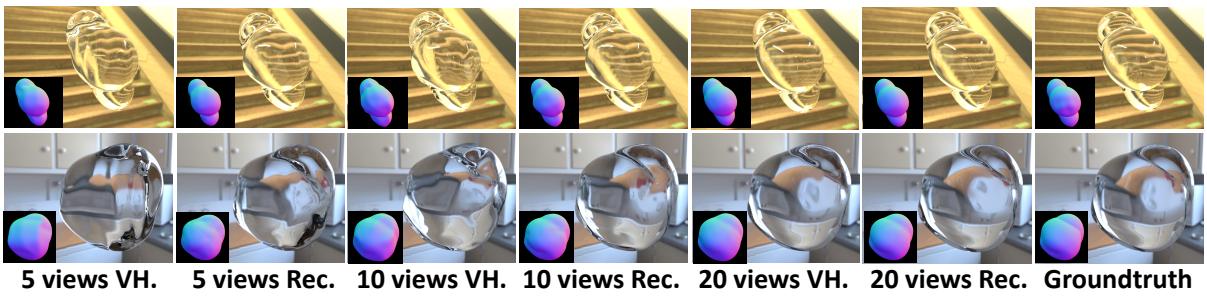



**Figure 5.17.** Transparent shape reconstruction in our synthetic dataset using 5, 10 and 20 views. Images rendered with our reconstructed shapes are much closer to the those rendered with ground truth shape, as compared to images rendered with the visual hull shapes. The inset normals are rendered from the reconstructed shapes and demonstrate the same conclusion.

**Table 5.6.** Quantitative comparisons of normal estimation from 5 and 20 views. Following the notation in this chapter, `vh5` and `vh20` represent the initial normals reconstructed from visual hulls corresponding to 5 and 20 views, respectively. Here, `wr` and `basic` are our basic encoder-decoder network with and without rendering error map ($I^{er}$) and total reflection mask ($M^{tr}$) as inputs. Further, `wr+cv` represents our network with cost volume and `wr+cv + opt` represents the predictions after optimizing the latent vector to minimize the rendering error. Similar to the 10-view case, `wr+cv + opt` performs better than all other baselines for transparent shape reconstruction using both 5 and 20 views.

| 5 views normal reconstruction | vh5 | basic | wr | wr+cv | wr+cv +op |
|---|---|---|---|---|---|
| $N^1$ median (°) | 12.7 | 6.1 | 6.0 | 6.0 | **5.9** |
| $N^1$ mean (°) | 15.3 | 7.8 | 7.9 | 7.8 | **7.7** |
| $N^2$ median (°) | 18.3 | 10.7 | 10.7 | 10.5 | **10.0** |
| $N^2$ mean (°) | 20.9 | 12.5 | 12.5 | 12.3 | **11.9** |
| Render Err.($10^{-2}$) | 9.7 | 5.9 | 5.8 | 5.9 | **4.1** |
| 20 views normal reconstruction | vh20 | basic | wr | wr+cv | wr+cv +op |
| $N^1$ median (°) | 2.5 | 2.2 | 2.2 | 2.2 | **2.2** |
| $N^1$ mean (°) | 4.6 | 3.4 | 3.4 | 3.3 | **3.3** |
| $N^2$ median (°) | 5.2 | 4.7 | 4.6 | 4.6 | **4.3** |
| $N^2$ mean (°) | 7.6 | 6.5 | 6.4 | 6.3 | **6.1** |
| Render Err.($10^{-2}$) | 4.0 | 3.7 | 3.8 | 3.8 | **2.7** |

the image formation process within the network can lead to better normal reconstruction results. Figure 5.16 shows two normal reconstruction results on our synthetic dataset. For both examples, our physically-based network performs significantly better than the classical visual hull method for 5, 10 and 20 views.

**Modification of standard PointNet++ [172]**

We examine our modifications of the standard PointNet++ architecture for point cloud reconstruction to better incorporate normal information. The quantiative numbers are summarized in Table 5.7. We first remove single modifications from standard PointNet++ to our novel version (– maxPooling, – normal Diff. and – normal Skip.) and then remove all the modifications to use standard PointNet++ to reconstruct the point cloud of our transparent shapes (standard). Experiments show that each of our modifications brings consistent improvements

**Table 5.7.** Comparisons of point cloud reconstruction with different PointNet++ architectures on our synthetic dataset. Following the notation in this chapter, RE represents rendering error based view selection. $\mathscr{L}_P^{\text{CD}}$ represents the Chamfer distance loss.

| | CD($10^{-4}$) | CDN-mean(°) | CDN-med(°) | Metro($10^{-3}$) |
|---|---|---|---|---|
| RE-$\mathscr{L}_P^{\text{CD}}$ | **2.00** | **6.02** | **4.38** | **5.98** |
| – maxPooling | 2.09 | 6.26 | 4.59 | 6.09 |
| – normal Diff. | 2.09 | 6.31 | 4.62 | 6.48 |
| – normal Skip. | 2.07 | 6.14 | 4.51 | 6.20 |
| standard | 2.12 | 6.34 | 4.72 | 6.49 |

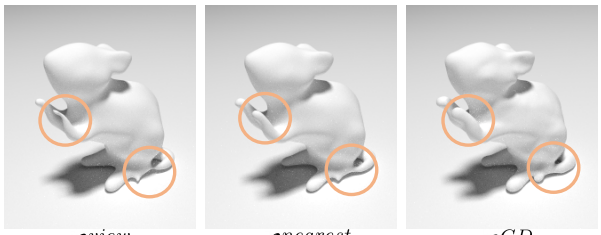



**Figure 5.18.** Comparisons of point cloud reconstruction with different loss functions on a real example. Our modified PointNet++ trained with Chamfer distance loss achieves better quality compared with the other two losses.

in reconstruction accuracy and removing all of them leads to a much poorer performance. This shows our modifications ease the difficulty for the network to reason about point cloud distribution based on normal predictions. Figure 5.18 demonstrates a real example reconstructed by our modified PointNet++ trained using different loss functions. It is clearly observed that our modified PointNet++ trained with Chamfer distance loss leads to a more complete and less noisy 3D reconstruction, especially for thin structures and concave regions.

### Optimization of latent vector

We adopt an alternating minimization strategy to optimize the latent vector. We first keep $N^1$ unchanged and only change $N^2$ by adding a large identity loss on $N^1$. After 500 iterations, we remove the constraint and optimize both $N^1$ and $N^2$ simultaneously. This is because the our $N^1$ prediction is usually more accurate and optimizing $N^2$ first can lead to better results. In Table 5.8, we compare the normal reconstruction results of optimizing the latent vector and directly

**Table 5.8.** Quantitative comparisons of different optimization strategies for normal estimation from 10 views. `op` represents the optimization of the latent vector. `opPixel` represents the optimization in the pixel space.

| 10 views normal reconstruction | `vh10` | `wr+cv +op` | `wr+cv +opPixel` |
|---|---|---|---|
| $N^1$ median (°) | 5.5 | **3.4** | 3.8 |
| $N^1$ mean (°) | 7.5 | **4.8** | 4.9 |
| $N^2$ median (°) | 9.2 | **6.6** | 7.4 |
| $N^2$ mean (°) | 11.6 | **8.4** | 8.5 |
| Render Err.($10^{-2}$) | 6.0 | 2.9 | **2.6** |


**Figure 5.19.** Appearance changes for same shape geometry under various index of refraction (IoRs). IoRs range from 1.3 to 1.7.

optimizing the per-pixel normals. The quantitative comparison shows that while optimizing per-pixel normal can also decrease the rendering error, only by optimizing the latent vector can we observe improvements in normal reconstruction accuracy. The inherent ill-posed nature of normal prediction of transparent shapes makes it necessary to have a strong regularization to obtain meaningful outputs. In this case, the regularization is provided by the trained decoder which constrains the predicted normals to be on the natural shape manifold.

### 5.4.5 Sensitivity Analysis for Index of Refraction

As mentioned in Section 5.4.1, we perform a sensitivity analysis on the influence of a different test-time IoR on the shape reconstruction accuracy. We re-render our synthetic transparent testing set with the same shapes and environment maps. However, instead of rendering with a fixed IoR value of 1.4723, we randomly sample 5 different IoRs ranging from 1.3 to 1.7 for each shape. Figure 5.19 shows an example of the same shape rendered under different IoRs. We then test our network trained with a fixed IoR value of 1.4723 on the new test

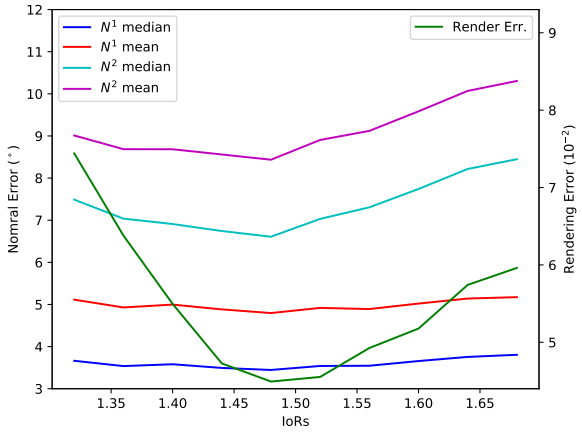



**Figure 5.20.** The mean normal estimation errors across varying IoRs in the test set, using the fixed training set IoR value for prediction.

set with variable IoR. During testing, the IoR used by the rendering layer is kept fixed at 1.4723. The quantitative comparisons have been summarized in Table 5.2 and 5.3.

Figure 5.20 and 5.21 show trends in the normal and shape reconstruction errors across varying IoRs in the test set. As expected, the errors are relatively smaller for IoRs close to the training set value of 1.4723. In particular, this trend is more explicitly visible in normal estimation, since the model leverages the features from the rendering layer and cost volume which require known IoRs. However, the overall variation in error is small across this range of IoRs.

The above plots further support the analysis in Table 5.2 and 5.3. Even though the predicted normals and the final reconstructed mesh are expectedly more accurate in the known IoR case, the quantitative errors increase gracefully and not too much across a range even with unknown IoR. This suggests that our network is relatively robust to the IoR value. Our future work will consider simultaneously reconstructing the transparent shape and predicting its IoR.

## 5.5 Discussion

We present the first physically-based deep network to reconstruct transparent shapes from a small number of views captured under arbitrary environment maps. Our network models the properties of refractions and reflections through a physically-based rendering layer and cost

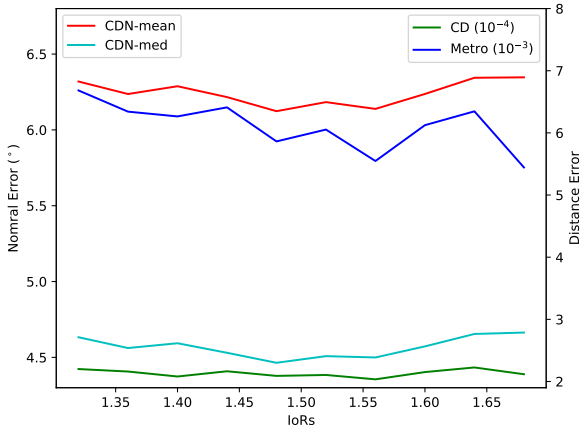



**Figure 5.21.** The mean shape reconstruction errors across varying IoRs in the test set, using the fixed training set IoR value for prediction.

volume, to estimate surface normals at both the front and back of the object, which are used to guide a point cloud reconstruction. Extensive experiments on real and synthetic data demonstrate that our method can recover high-quality 3D shapes.

**Limitations and future work**

Our limitations suggest interesting future avenues of research. A learnable multiview fusion might replace the visual hull initialization. We believe more complex light paths of length greater than 3 may be handled by differentiable path tracing along the lines of differentiable rendering [120, 262]. While we assume a known refractive index, it may be jointly regressed. Finally, since we reconstruct $N^2$, future works may also estimate the back surface to achieve single-view 3D reconstruction.

Chapter 5 is based on the material as it appears in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020 (”Through the Looking Glass: Neural 3D Reconstruction of Transparent Shapes”, Zhengqin Li*, Yu-Ying Yeh*, Manmohan Chandraker). The dissertation author was one of the primary investigators and authors of this paper. Zhengqin Li is a co-first author who contributed equally to the paper.

# Chapter 6

# Learning to Relight Portrait Images via a Virtual Light Stage and Synthetic-to-Real Adaptation

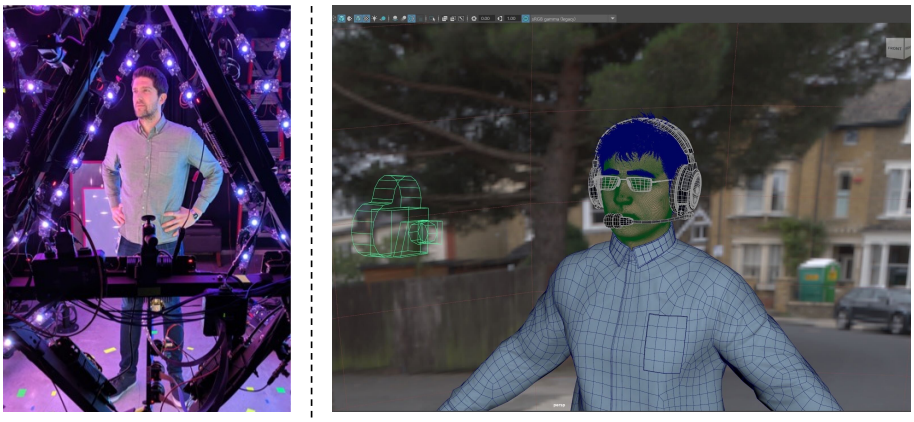
## 6.1 Introduction

Given a photograph of a person, portrait relighting aims to re-illuminate the person as if they appeared in a new lighting environment. While this has been long used in professional settings, the growing interests in smartphone photography and video conferencing tools create a need for casually enhancing [156, 270] and re-illuminating a person from an in-the-wild image or video [160]. Relighting a person as if they really belong to the scene requires knowledge of the 3D scene geometry and materials to simulate complex lighting interactions, including global illumination. Therefore, portrait relighting from a 2D in-the-wild portrait without any prior knowledge is extremely challenging. Due to the under-constrained nature of the problem, previous work relied on 3D face priors [205, 45], intrinsic images and shape from shading [14, 197, 93, 80], or style transfer [203]. However, these models often fail to handle complex skin surfaces and subsurface reflectance, wide varieties of clothes or hairstyles, or complex inter-reflections between eyeglasses and other accessories. Capture-based data-driven techniques [41, 236, 239] have been proposed to achieve realistic relighting of human faces, but every subject needs to be recorded by a complex capture rig, limiting their applicability. Recently, the most


**Figure 6.1.** Top: Given an input portrait image and a target high dynamic range environment map (rendered as a diffuse ball and a mirror ball in the inset), our relighting method generates relit results under different illuminations with high photorealism. This is achieved by learning relighting physics from a synthetic dataset rendered using a virtual light stage, as well as adapting the learned model with a real-world dataset to close the domain gap. Bottom: our framework also enables new features, such as controlling the glares on glasses (diffuse and mirror balls shown in the inset), as well as synthesizing more temporally-consistent relit videos.

successful methods for in-the-wild portrait relighting are based on deep learning. They train deep neural networks via a supervised framework using high-quality ground truth data captured by a light stage.

A light stage [41, 40, 65] is a spherical lighting rig equipped with hundreds or thousands of programmable inward-pointing light sources evenly distributed on the sphere. As each light source provides a unique lighting direction, it is useful to capture an object illuminated by one-light-at-a-time (OLAT). The OLAT images can be used as point-based lighting bases to realistically relight faces under novel environments via image-based relighting [41, 236]. Several prior portrait relighting methods [216, 160, 263] have used this technique to create the dataset necessary for training their portrait relighting networks. Each sample in the dataset consists of pairs of relit portrait images and their corresponding target environment maps. The network is



Physical Light Stage
[Guo et al. 2019]

Virtual Light Stage

**Figure 6.2.** Illustration of a virtual light stage versus a physical light stage. Our virtual light stage setup renders high dynamic range images from a virtual camera. We use HDR environment maps as skydome lighting and randomly select one subject from our 3D scan collections paired with randomly selected hairstyles, clothes, and accessories.

then trained to predict a relit image given an input portrait and a target environment map. While valuable for ground truth computation, constructing a light stage is an expensive task. Moreover, building a light stage is only the first step. One still needs to find many subjects to sit still inside the light stage for data collection. Capturing and processing the resulting OLAT scans also requires time-consuming and labor-intensive efforts.

To challenge the conventional wisdom of the light stage data being the necessity of achieving high-quality portrait relighting, we develop a new approach that can achieve comparable results with the state-of-the-art methods without relying on a physical light stage. Our approach is based on the key observation that we can decouple the relighting problem into two: 1) learning to approximate the physically-based lighting behaviors from synthetic data produced by a path tracer and 2) learning to synthesize photorealistic portrait images.

We tackle the first part by leveraging advancement in physically-based rendering [168]. Specifically, we build a virtual light stage to re-illuminate pre-captured photogrammetry face scans to construct the training dataset using a physically-based renderer, as shown in Figure 6.2. Using the virtual light stage provides several key advantages over the physical one. First, we

only need workstations to render a ground truth training dataset, saving the cost and hassle of building a physical one. Second, since photogrammetry-based face scans are much easier to acquire than OLAT scans, we are able to create a more diverse dataset with a higher diversity of ages, genders, and ethnicity. Finally, rendering a synthetic dataset also provides the ground truth for several other useful attributes. We show that this can enable new capabilities, such as controlling glares in eyeglasses in Section 6.5. We will release our synthetic dataset upon publication to support other researchers in conducting portrait relighting research.

Although the virtual light stage provides a way to learn the physically-accurate relighting behaviors from synthetic data, the realism of the deep network output is often limited when trained with only synthetic data. After all, the photogrammetry face scan comes with only basic reflectance information such as diffuse albedo and does not provide all the necessary information for realistic portrait relighting, such as surface [63, 155] and subsurface [91] reflectance properties. As a result, the generated synthetic training data exhibits a domain gap from their real counterparts. To close this gap, we propose a synthetic-to-real adaptation approach. We use in-the-wild portrait images to adapt the intermediate representations learned in our portrait relighting network for achieving photorealistic outputs. A specific challenge for the adaptation is how to improve photorealism while still maintaining the ability to relight an image coherently with the provided lighting condition. To this end, we propose self-supervised lighting consistency losses that promote the network to maintain consistent relighting during domain adaptation. We show that our approach leads to competitive portrait relighting performance to the state-of-the-art.

We further extend our relighting framework to video portrait relighting. Instead of applying relighting in a frame-by-frame manner, which often leads to flickering artifacts, we model the extension from image to video as a domain adaptation problem, following the same spirit of our synthetic-to-real approach. We use in-the-wild videos to adapt the intermediate representations learned in our portrait relighting network to improve temporal consistency. This results in a video portrait relighting network that is useful for various video applications.

In summary, our contributions include the following:

- We propose a learning-based portrait relighting method that can achieve state-of-the-art results for challenging in-the-wild photos without requiring training data from a light stage.
- Our approach learns to approximate the physics for portrait relighting from a synthetic dataset, carefully constructed using a physically-based rendering engine. We further propose a novel synthetic-to-real adaption approach that leverages in-the-wild photos and a set of self-supervised lighting consistency losses to achieve photorealistic outputs.
- We extend our approach to portrait video relighting. We show that videos relit by our framework contain much less flickering and look temporally smoother than existing video relighting approaches.

The chapter is organized as follows. We review related works in Section 6.2. Section 6.3 presents our training data generation via a virtual light stage. Section 6.4 gives details of our relighting network design and its adaptation from synthetic to real and from images to videos. Experiment results are included in Section 6.5. Finally, Section 6.6 concludes the chapter.

## 6.2 Related work

### 6.2.1 Portrait Relighting

Debevec *et al.* [41] show portrait relighting can be achieved by projecting the environment map to the reflectance field constructed by the OLAT images captured with a light stage setup. Wenger *et al.* [236] further extend the framework to allow moderate subject motions during the OLAT image acquisition. These OLAT-based approaches can produce high-quality relighting results for the subject whose OLAT capture is available but cannot be applied to unseen subjects. Despite the limitation, these methods are valuable for building the ground truth dataset for learning-based portrait relighting methods discussed below.

Various deep neural network architectures exist for portrait relighting based on directional

lighting [157], and HDR environment map lighting [216, 160, 135, 263]. SIPR-S [216] uses an encoder-decoder architecture. Total Relighting (TR) [160] combines a set of UNets to generate the final output and achieves state-of-the-art results. Zhang *et al.* [263] propose a network architecture for video portrait relighting. NLT [267] learns a 6-DoF light transport function of UV locations, lighting directions, and viewing directions to enable novel view synthesis and relighting from OLAT images. Our architecture is based on the TR framework, but we have made several improvements. First, we modify it to leverage supervision from our own custom synthetic data. We also add novel modules to bridge the synthetic-to-real domain gap and achieve temporally consistent video portrait relighting results.

Similar to ours, SIPR-W [234] uses captured face scans to create synthetic data to train portrait relighting networks. However, their synthetic humans have fixed looking. They do not pair them with different hair, clothes, and accessories. As a result, the diversity of the training data is limited, which throttles the relighting network performance. Moreover, they do not handle the synthetic-to-real domain gap and fall short in achieving photorealistic outputs. More recently, Sengupta *et al.* [195] propose a lightweight capture system to record facial appearance using only an LED monitor at the desk, circumventing the need for a light stage. However, their monitor-based capture setup constrains their method to limited brightness controls and head motions, and near and frontal light sources. Other deep learning-based portrait relighting methods exist that do not require high-quality ground truth data. Still, they assume spherical harmonics (SH) lighting [197, 272, 81] that limit their method to low-frequency Lambertian surfaces. Our method adapts HDR image-based relighting representation that can handle arbitrary natural environments, including specular highlights and cast shadows.

### 6.2.2 Learning with Synthetic Datasets

Synthetic data provide valuable supervision for tasks where acquiring ground truth real data is challenging. They have been shown powerful for various tasks, including pose estimation [204], face analysis [240], pedestrian detection [51], segmentation [185], inverse

rendering [121, 126], 3D reconstruction [125, 241], single-image full-body relighting [94, 113] and scene relighting [169, 170]. Our work is along this line. We show synthetic data are useful in producing state-of-the-art relighting results.

One drawback of using synthetic data to train a learning-based approach is the exposure to the synthetic-to-real domain gap problem. After all, synthetic data and real data follow two different distributions. To bridge the gap, many prior works develop methods that leverage domain knowledge in their applications. Representative approaches include synthetic-to-real for semantic segmentation [48, 79], 3D human pose estimation [46], animal pose estimation [117], GTA-to-real [184], SVBRDF estimation [226], depth estimation [8, 271], vehicle re-Id [115], and multi-view portrait view synthesis and relighting [217]. We follow the same spirit and devise a synthetic-to-real approach dedicated to the single-view portrait relighting task. In a closely related work, Tajima *et al.* [218] propose a two-stage method for single-image full-body relighting with synthetic-to-real domain adaptation. In the first stage, they train a neural network for diffuse-only relighting using a spherical harmonic-based lighting representation. In the second stage, they train a network for enhancing non-diffuse reflection by learning residuals between real photos and images reconstructed by the previous-trained diffuse-only network. Our method differs from theirs in several aspects. First, we focus on portrait images that contain high-resolution facial details where errors are less forgiving. Second, our method takes environment maps as the lighting representation for image-based relighting, which is more expressive. Third but not least, we develop a novel self-supervised lighting consistency loss to achieve high-quality results.

## 6.3 Synthetic Dataset Generation

For training a portrait relighting network, paired data are required for supervision. In the dataset, each sample consists of a pair of images of the same person at the same pose and expression but under two different lighting conditions. We generate the paired training dataset

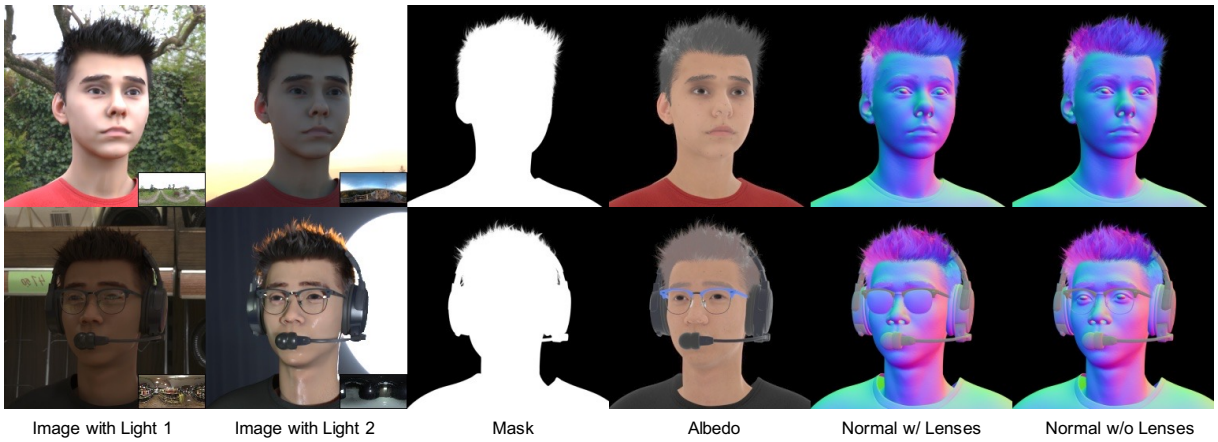



**Figure 6.3.** Examples of our dataset rendering. Our rendering consists of paired images under different lighting using HDR maps (as shown in the bottom right insets), along with the corresponding foreground mask, albedo, and normal maps. In addition, we render normal maps with lenses and without lenses to better model the reflections on glasses. Additional rendered attributes such as diffuse, specular, and shadow maps can be found in Figure 6.5.

**Table 6.1.** Dataset comparisons for different portrait relighting methods. SIPR-S [216], TR [160] and NVPR [263] use a light stage to capture OLAT sequences and generate relit ground truth images. SIPR-W [234] and Ours acquire a large number of 3D face scans and render synthetic datasets by applying HDR maps as the environment lighting.

| | Data acquisition | Subjects | | Add-ons per subject | | | HDR maps | |
|---|---|---|---|---|---|---|---|---|
| | | Quantity | Genders | Hairstyles | Clothes | Accessories | Real | Synthetic |
| SIPR-S | light stage | 22 | 17M, 5F | single | single | single | 3094 | × |
| TR | light stage | 70 | - | single | multiple | multiple | 200 | × |
| NVPR | light stage | 36 | 18M, 18F | single | single | single | 2810 | × |
| SIPR-W | simulation | 108 | - | none | none | none | 391 | × |
| Ours | simulation | 515 | 247M, 265F | multiple | multiple | multiple | 1606 | ✓ |

with a virtual light stage and a synthetic human generation pipeline described below.

Our synthetic human generation pipeline leverages pre-captured face scans that contain various identities in different facial expressions. Note that these face scans come with neither hair nor accessories. We thus randomly pair them with strand hair models of different styles and with various accessories and clothes. Finally, we put the synthetic humans in a virtual light stage where we illuminate them using HDR environment maps. The virtual light stage rendering is performed using a physically-based path-tracing renderer to simulate the light transport faithfully.

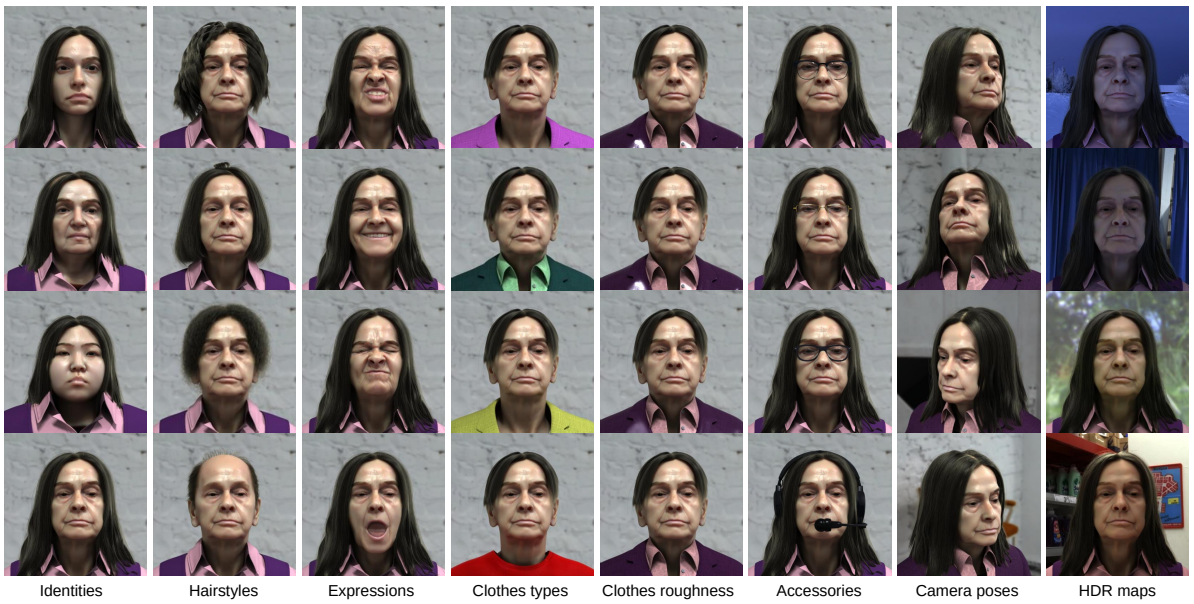



**Figure 6.4.** Examples of the variations in our dataset rendering. For each column, we change one variable which is specified at the bottom.

### 6.3.1 3D Face Scans

We use high-quality 3D photogrammetry-based face scans with high-resolution texture maps for our synthetic humans. We note that these assets can be captured through a relatively low-cost system [15] based on a passive camera array, which is much cheaper than a light stage. Moreover, there are already commercially available high-quality 3D face scans. The acquisition cost is lower than building a physical light stage. We acquired a commercial 3D face scan collection from the Triplegangers website[1]. Each face scan is associated with a high-quality 4k diffuse texture and a high-resolution face geometry to represent mesoscopic facial details. Since the output mesh from a multi-view stereo process is unstructured, we fit our template model of common topology with 12325 vertices to the scan with non-rigid ICP and represent the remaining facial geometry details using a 4K displacement map. This common template registration is necessary to automate the synthetic data generation processes, such as fitting various hairstyles, applying common rigid transformations, and apply common UV-space

[1] https://triplegangers.com/

material maps. We use the Arnold *aiStandardSurface*[2] shader to describe the face material, which includes path-traced subsurface scattering and specular highlights. We set the diffuse texture as the subsurface scattering color. We also apply different materials to different eye regions to render the specularity of eyes. Details of face materials can be found in Section 6.3.5.

The Triplegangers dataset consists of 248 males and 267 females of unique identities. Each identity has up to 20 facial expression variations, leading to over 10k face scans in total. It also features a diverse range of ages and ethnicity, including ages ranging from 18 to 76-year-old and six ethnic groups. The detailed statistics can be found on the Triplegangers website.

### 6.3.2 Hair

Since our face scans do not contain hair, we use XGen [223] in Maya[3] to model our own custom hair strand templates. Each hairstyle is created by an experienced artist and is composed of several XGen descriptors for different hair portions, such as eyelashes, eyebrows, and beards. The Arnold aiStandardHair[4], a physically-based hair shader, is applied to these XGen descriptors. In total, we have 24 hairstyles, and each of them can be branched into many variations. To further increase the diversity, we uniformly sample the *melanin* (within (0, 1)) and *melanin redness* (within (0, 1)) to change hair colors. In our dataset, each hairstyle is fitted to various head scalps of 3D face scans.

### 6.3.3 Clothing and Accessories

We acquire seven 3D garment models from the Turbosquid website[5] to add clothing variations to our synthetic humans. These seven garments have different styles, ranging from business shirts to T-Shirts. In addition, we randomize the color and the roughness of clothes (from 0.1 to 1.5) to cover a wider range of clothing.

[2]https://docs.arnoldrenderer.com/display/A5AFMUG/Standard+Surface
[3]https://www.autodesk.com/products/maya/
[4]https://docs.arnoldrenderer.com/display/A5AFMUG/Standard+Hair
[5]https://www.turbosquid.com

We also acquire three different types of eyeglasses and one headphone from the Turbosquid website. These accessories are randomly added to our synthetic humans during rendering. These assets are added since they are most common for video conferencing, which is one of the major applications for single-image portrait relighting. In the future, we plan to add other accessories such as earring, necklace, scarf, and mask.

### 6.3.4 Rendering

We use Arnold [60] in Maya, a physically-based path tracer, to render our training dataset. The renderer simulates the light transport based on physics and can create realistic outputs.

The training dataset consists of paired samples. Each sample consists of a synthetic human with the same pose and expression rendered under two different lighting. We create a synthetic human by sampling a face scan, a hairstyle, a garment, a pair of eyeglasses (at a probability of 0.2), and a headset (at a probability of 0.05). The synthetic human is positioned near the world center with some randomness in the precise location. We sample the camera pose from a sphere centered at the world center with a fixed radius. The viewing angle is restricted to within a range of $(-45°, 45°)$ horizontally and $(-22.5°, 22.5°)$ vertically in front of the faces. The camera pose variations result in variations in head poses in the rendered images of the synthetic humans.

Once a synthetic human is created, and a camera pose is sampled, we use two different environment maps to illuminate it to create the paired sample. We use latitude-longitude format HDR images as our lighting. We acquire 1536 HDR images for training and 70 images for evaluation from a variety of sources, including Poly Haven[6]. The HDR images are captured from various environments, including indoors and outdoors. To add more lighting variations, we apply random rotations and horizontal flips to the HDR images.

In addition to color images, we also render other attributes such as albedo and normal maps, which can be easily computed using our synthetic data generation pipeline. Moreover,

[6]https://polyhaven.com

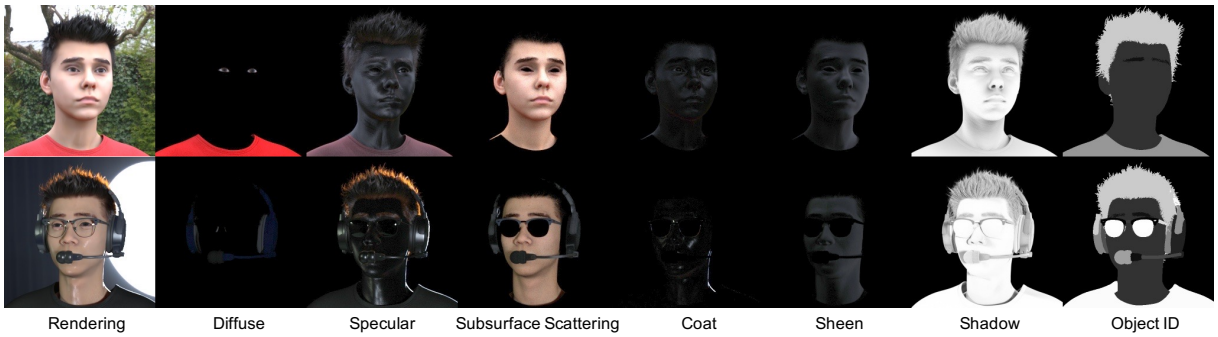



**Figure 6.5.** Example of additional rendering for implicit components. Diffuse, specular, subsurface scattering, coat and sheen are used in Arnold aiStandardSurface for face materials. We also explicitly rendered shadow maps and object ID maps which might be useful for future research.

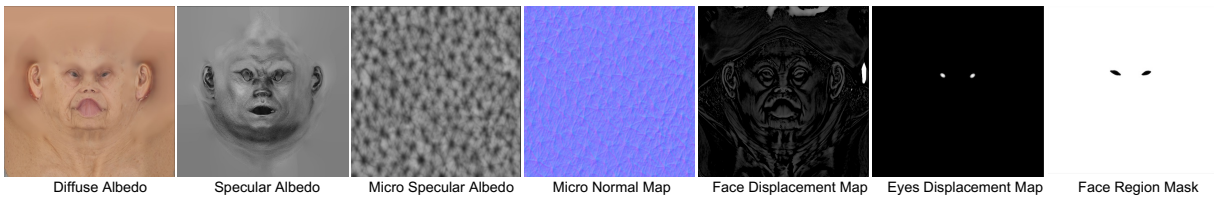



**Figure 6.6.** Example of spatially-varying maps used in Arnold *aiStandardSurface* shader.

we render two versions of surface normals, one with lenses in eyeglasses and one without. The double normal map treatment helps us synthesize glares in the eyeglasses. Examples of rendered results can be found in Figure 6.3. We demonstrate variations of hair, identities, expressions, clothes, and accessories available to a face scan in our dataset in Figure 6.4. Overall, we rendered about 300k paired samples, where each sample contains a pair of RGB images and the attributes mentioned above. Each image has a resolution of $512 \times 512$ and is stored in the HDR format.

### 6.3.5 Additional Details on the Synthetic Dataset

In addition to the rendering example shown in Figure 6.3, our synthetic data generation pipeline can generate other intrinsic components rendered but unused in our framework. As shown in Figure 6.5, we can also render diffuse, specular, subsurface scattering, coat, and sheen components which all rendered from the Arnold *aiStandardSurface* shader. We will release our synthetic dataset.

As shown in Figure 6.6, we use a mask in the texture space to control different parameters

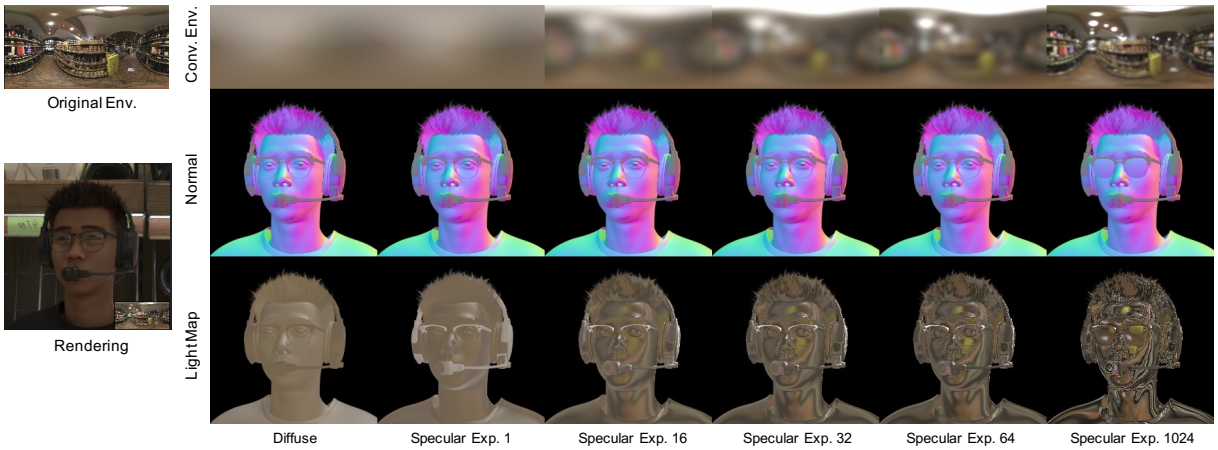



**Figure 6.7.** Example of light maps. We compute prefiltered environment maps by diffuse and specular convolution. We then compute diffuse light map $L_d$ and specular light maps $L_s^1$, $L_s^{16}$, $L_s^{32}$, $L_s^{64}$ by indexing the prefiltered maps using normal without glasses $N$ and using normal with glasses $N_l$ to compute $L_{s,l}^{1024}$.

for face and eye regions. For face regions, we apply 0.95 weight to subsurface scattering (SSS) and set the RGB texture (diffuse albedo) as SSS color, (1, 0.35, 0.2) as RGB Radius, 0.225 as scale, *randomwalk_v2* as type, and 0 as anisotropy. We apply a specular albedo map to specular weight. Then, we set 0.82 to specular color and 0.4 to specular roughness, respectively. We set 0.1 to coat weight, a micro-specular albedo map (UV repeated 160 times) to coat color, 0.25 to coat roughness, and 1.5 as coat IOR. We set 0.1 as sheen weight, 1.0 as sheen color, and 0.3 as sheen roughness. We apply a micro-normal map (UV repeated 160 times) as a bump map. We apply a displacement map (computed from a raw 3D face model) to skin regions.

The eye and inner mouth regions of our models are represented by a billboard mesh that connects upper and lower eyelids or lips vertices with a projected texture similar to [130].

For eye regions, we set the RGB texture (diffuse albedo) as the base color. We apply a gradient map (0.2 at the center, 1.0 at the boundary, computed from $1 - 0.8 \cdot$ *eye displacement* below) to two eyeball regions of the diffuse texture to compensate for the unwanted refection baked from the data capture. Specular weight and color are the same as the face, but we set 0.1 for specular roughness. We set 0.5 to coat weight, the same micro-specular albedo map for coat color, 0.1 for coat roughness, and 1.376 for coat IOR. The same micro-normal map is applied

as a bump map. We also set a gradient map as the displacement map to simulate the eyeball geometry which is otherwise flat in the original face model. There are no SSS and sheen applied to eye regions. In addition to the components from the shader, we also render shadow maps and object ID maps, which might be useful for future research.

We also demonstrate the light maps computed from the input HDR environment map and normal in Figure 6.7.

## 6.4 Method

The full framework of our portrait relighting method is shown in Figures 6.8 and 6.9. We first obtain the foreground image $I$ using an off-the-shelf matting network [103] given an input portrait image. This image and an environment map are inputs to our relighting network to predict a relit image $R$.

Our relighting network is based on the TR framework [160]. However, instead of adopting it directly, we made several modifications to adapt it to our scenario. Below we first briefly review the TR framework (Section 6.4.1). We then explain how we modify it to train on our synthetic dataset (Section 6.4.2). Next, we finetune the model on real images to bridge the synthetic-to-real domain gap (Section 6.4.3). Finally, we show how we can further finetune the model on real videos to increase temporal stability when relighting portrait videos (Section 6.4.4).

### 6.4.1 Total Relighting [160]

At a high level, the TR framework works as follows: Given an image, it first predicts the normal and albedo maps. It then predicts the diffuse and specular light maps with respect to the target environment map illumination using the predicted normals. Finally, these components are concatenated as input to another network to produce the final relit image. The framework consists of 4 networks: a geometry network $f_N^{TR}$, an albedo network $f_A^{TR}$, a specular network $f_S^{TR}$, and a rendering network $f_R^{TR}$. These networks are all implemented using a UNet architecture.

The geometry network $f_N^{TR}$ first takes the foreground image $I$ as input, and generates a

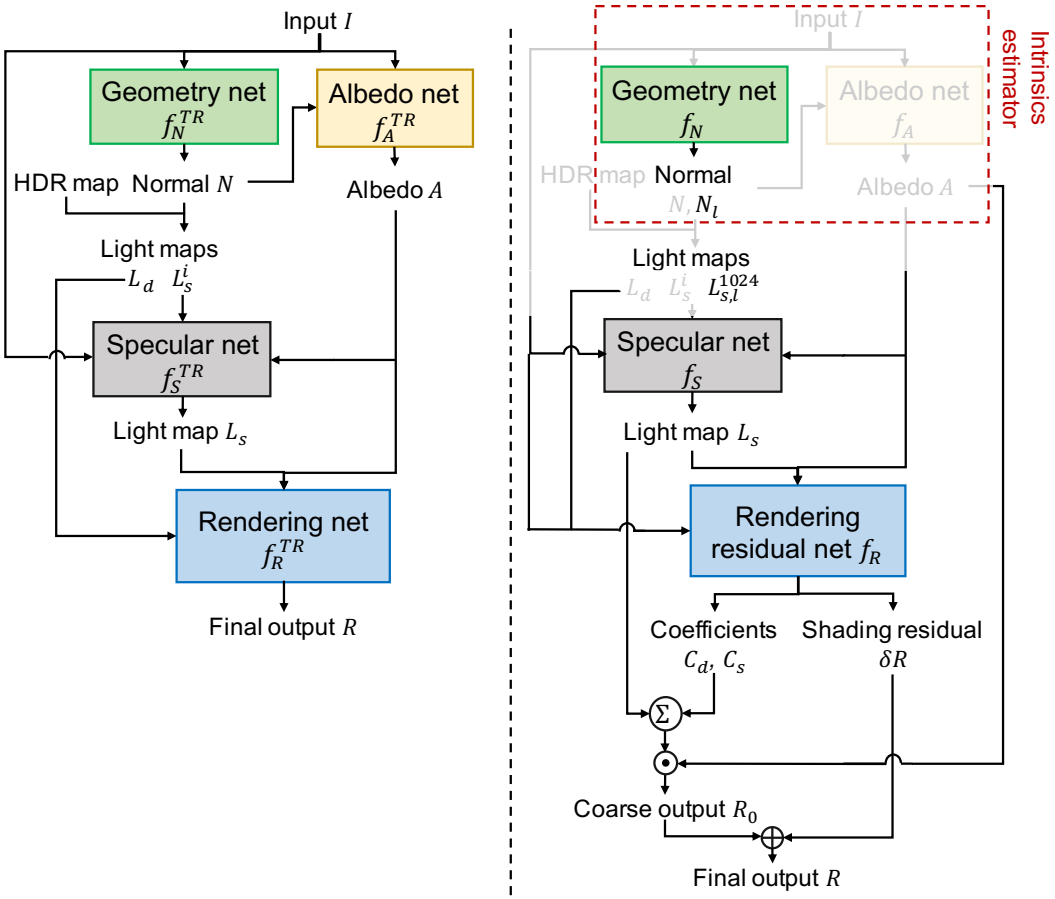



**Figure 6.8.** Our portrait relighting method when trained on the synthetic dataset. (a) The baseline Total Relighting network [160]. (b) Compared with (a), we make two modifications. First, we predict an additional normal map $N_l$ which contains glasses lenses, so it can be used to create a new highly specular light map $L_{s,l}$ for glares on glasses. Second, instead of using $f_R$ as a black box to generate the final output, we first obtain a coarse output $R_0$ using the estimated albedo and light maps. We then only estimate the fine details $\delta R$ to be added to this coarse output to obtain the final output. Note that the components similar to (a) are grayed out.

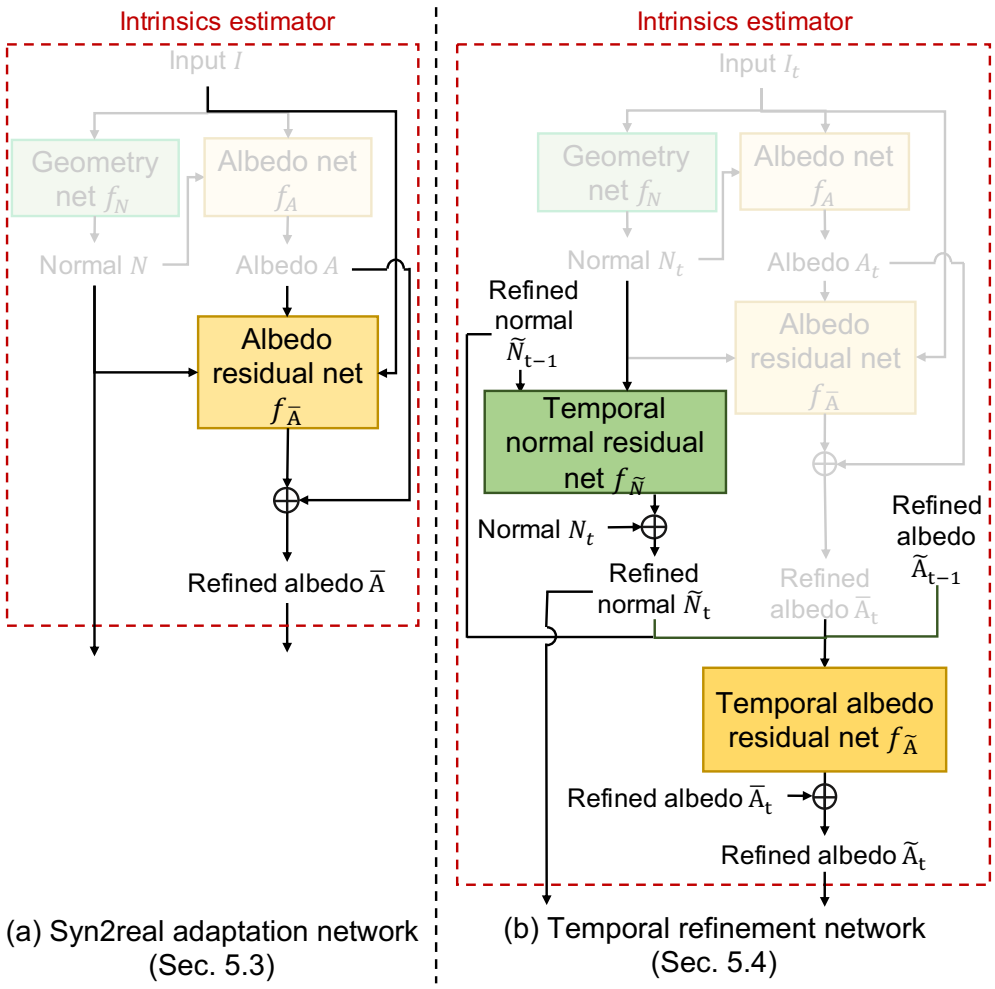



**Figure 6.9.** Our syn2real adaptation and temporal refinement networks. Note that we only modify the intrinsics estimator block in Figure 6.8. (a) To bridge the domain gap between synthetic and real domains, we observe that the gap mainly comes from the estimated albedo map. We thus train a domain adaptation network to refine the albedo map by predicting an albedo residual, using only *unpaired* real data. (b) Finally, to increase temporal stability when relighting videos, we can (optionally) train two more residual networks to refine the normal and albedo maps using real videos. For each subfigure, the part similar to the previous subfigure is grayed out.

**Table 6.2.** Notations used in this chapter.

| Baseline | | Syn2real | | Temporal | |
|---|---|---|---|---|---|
| $I$ | Input foreground image | | | | |
| $N$ | Normal | $N_l$ | Normal with lenses | $\widetilde{N}_t$ | Temporally-refined normal |
| | | | | $\delta\widetilde{N}_t$ | Temporally-refined normal residual |
| $A$ | Albedo | $\bar{A}$ | Refined albedo | $\widetilde{A}_t$ | Temporally-refined albedo |
| | | $\delta\bar{A}$ | Refined albedo residual | $\delta\widetilde{A}_t$ | Temporally-refined albedo residual |
| $L_d$ | Diffuse light map | | | | |
| $L_s$ | Specular light map | $L_{s,l}$ | Specular light map w/ lenses | | |
| $W_s$ | Weight for specular light map | $W_{s,l}$ | Weight for specular light map w/ lenses | | |
| $R$ | Relit output image | $\bar{R}$ | Refined relit output | $\widetilde{R}_t$ | Temporally-refined relit output |
| $R_0$ | Initial Rendering | | | | |
| $\delta R$ | Relit image residual | | | | |
| $f_N$ | Normal net | | | $f_{\widetilde{N}}$ | Temporally-refining normal net |
| $f_A$ | Albedo net | $f_{\bar{A}}$ | Albedo refining net | $f_{\widetilde{A}}$ | Temporally-refining albedo net |
| $f_S$ | Specular net | | | | |
| $f_R$ | Render net | | | | |

surface normal map $N$,

$$N = f_N^{TR}(I). \tag{6.1}$$

Next, the foreground image $I$ is concatenated with the surface normal map $N$ as input and sent to the albedo network $f_A^{TR}$. The output is an albedo map $A$,

$$A = f_A^{TR}(I,N). \tag{6.2}$$

For environment maps, a diffuse irradiance map and a set of prefiltered environment maps according to specular Phong exponents ($n = 1, 16, 32, 64$) are precomputed [146]. The prefiltering of environment maps makes the integration computation offline and thus enables the real-time rendering of diffuse and specular materials. Here the *light maps* $L_d, L_s^1, L_s^{16}, L_s^{32}, L_s^{64}$ can be computed by simply indexing the prefiltered maps using the normal or reflection directions depending on predicted surface normals $N$. The diffuse and specular light maps are more efficient representations of lighting than the original environment map as they now embed the diffuse and specular components of illumination in the pixel space.

The specular network $f_S^{TR}$ takes foreground image $I$, albedo map $A$, and four specular light maps $L_s^1, L_s^{16}, L_s^{32}, L_s^{64}$ as input, and predicts the four-channel per-pixel weight map

$W_s^1, W_s^{16}, W_s^{32}, W_s^{64}$,

$$W_s^1, W_s^{16}, W_s^{32}, W_s^{64} = f_S^{TR}(I, A, L_s^1, L_s^{16}, L_s^{32}, L_s^{64}). \tag{6.3}$$

The weighted specular light map $L_s$ can be computed by $L_s = \sum_i W_s^i \odot L_s^i$, where $i = 1, 16, 32, 64$ and $\odot$ indicates pixel-wise multiplication.

Finally, the rendering network $f_R^{TR}$ takes albedo map $A$, diffuse light map $L_d$, and specular light map $L_s$ as input and predicts the relit image $R$,

$$R = f_R^{TR}(A, L_d, L_s). \tag{6.4}$$

Pandey *et al.* [160] train the networks using $L_1$ losses, perceptual losses, and GAN losses on the normal, albedo, and specular maps, as well as the final output. Please refer to their paper for more details.

### 6.4.2 Learning with Synthetic Data

We make two major modifications to the TR framework when training it on our dataset. The first one is that we improve the rendering network through a decomposition scheme. Unlike previous work that employs the rendering decomposition (e.g., [267, 156], we decompose the output to two *learnable* components: coarse rendering by predicting a set of parameters for a fixed rendering function and fine-scale residuals. The second modification is that we predict the normal for the lenses of the glasses prediction.

**Improved rendering network**

Different from the TR framework, which treats the rendering network $f_R$ as a black box to get the final relit image, we first compute an initial rendering from the predicted albedo, diffuse, and specular light maps. We then refine the result by only estimating the details which cannot be modeled by this coarse rendering. Specifically, the final rendering is decomposed into two

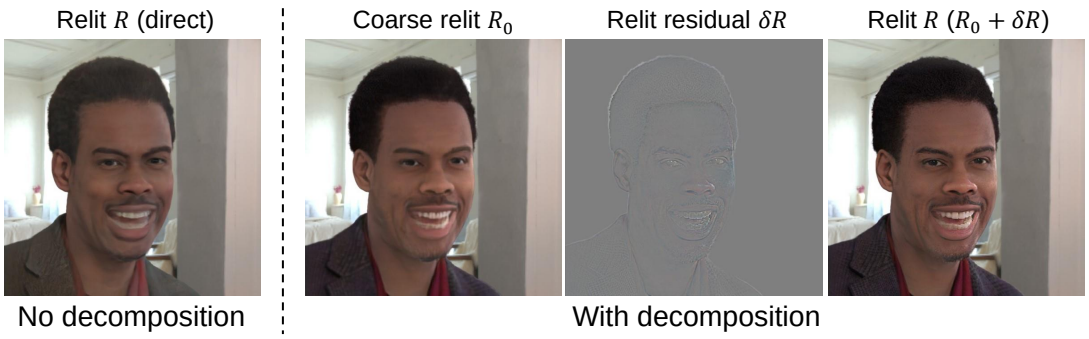



**Figure 6.10.** An example showing using a blackbox for final rendering vs. using an initial rendering plus adding fine details (e.g., teeth) as in our framework.

steps. We first predict coefficients to linearly combine the diffuse and specular light maps and then multiply the combined map by the albedo map to obtain an initial result,

$$R_0 = A \odot (C_d \cdot L_d + C_s \cdot L_s), \tag{6.5}$$

where $\odot$ stands for element-wise multiplication, and $C_d$, $C_s$ are the predicted 3-channel coefficients. We then estimate a residual map $\delta R$ to account for the details not modeled by this initial rendering. The final relit image can then be computed by $R = R_0 + \delta R$. An example of our decomposition of initial coarse rendering $R_0$, residual image $\delta R$, and the final relit image $R$ is shown in Figure 6.10. The facial details cannot be retained well if the final rendering is predicted by a blackbox without decomposition.

In our framework, the coefficients and the residual map are predicted by $f_R$, which takes the foreground image $I$, the albedo map $A$, the diffuse light map $L_d$, and the specular light map $L_s$ as input,

$$C_d, C_s, \delta R = f_R(I, A, L_d, L_s). \tag{6.6}$$

Similar to the TR framework, we employ losses on the normal, albedo, and specular maps, as well as the final output. In addition, to avoid the residual map $\delta R$ becoming too large and completely dominating the coarse rendering output $R_0$, we add an $L_1$ regularization on $\delta R$, i.e., $\mathscr{L}_{\delta R} = ||\delta R||_1$.

Example results by applying this modification are shown in Figure 6.21. Although this does not lead to large-quality improvement when trained on the synthetic dataset, it has significant effects when applying the synthetic-to-real adaptation to real data, as introduced in Section 6.4.3.

**Normal prediction for glasses**

Since our synthetic dataset contains normal maps both with lenses and without lenses for glasses, we train our network to predict both of them. In particular, our geometry network $f_N$ generates two normals,

$$N, N_l = f_N(I), \tag{6.7}$$

where $N$ is the normal map without lenses and $N_l$ is the normal map with lenses. If input has no eyeglasses, we predict the same normal maps for $N$ and $N_l$. Then, when generating the specular light maps, in addition to the four original specular exponents $(L_s^1, L_s^{16}, L_s^{32}, L_s^{64})$ using $N$, we generate an additional highly specular one $L_{s,l}^{1024}$ using $N_l$ to account for the glares. Our specular network $f_S$ then predicts an additional weight $W_{s,l}^{1024}$ compared to Equation. 6.3,

$$W_s^1, W_s^{16}, W_s^{32}, W_s^{64}, W_{s,l}^{1024} = f_S(I, A, L_s^1, L_s^{16}, L_s^{32}, L_s^{64}, L_{s,l}^{1024}). \tag{6.8}$$

The weighted specular light map then becomes $L_s = \sum_i W_s^i \odot L_s^i + W_{s,l}^{1024} \odot L_{s,l}^{1024}$ where $i = 1, 16, 32, 64$. This allows us to enable the eyeglasses glare controlling feature, as demonstrated in Figure 6.1, Figure 6.22, and the supplementary video[7].

### 6.4.3 Synthetic-to-Real Adaptation on Real Data

Although the model trained in Section 6.4.2 using our synthetic dataset can generate reasonable relit results, the outputs often lack photorealism when tested on real images. This is because there is a domain gap between rendered and real-world portrait photos. To bridge this gap, we propose a novel synthetic-to-real (syn2real) mechanism for portrait relighting.

[7] https://research.nvidia.com/labs/dir/lumos/

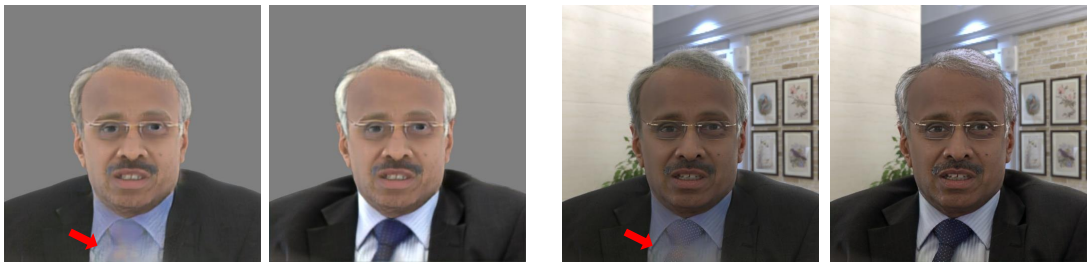



**Figure 6.11.** An example showing the initial vs. refined albedo and their corresponding relit outputs. Note how the refined albedo and relit output generate much better clothing.

In particular, instead of naively applying an existing syn2real network on the final relit image, we observed that the domain gap between synthetic and real data mainly comes from the diversity of the subject albedo, which strongly determines the final appearance in the relit image. People have more diverse hairstyles, face colors, minor facial marks, wearings, and accessories in the real world, which are missing in the synthetic data. This causes the albedo network trained with only synthetic data to fail when modeling real-world appearances. We tried a few variants of the adaptations including the adaptation of surface normals, but found empirically that the proposed albedo adaptation framework works better with our synthetic dataset.

To handle the domain gap between the albedo maps, we add an additional albedo residual network $f_{\bar{A}}$ to refine the initially estimated albedo. The network $f_{\bar{A}}$ takes the portrait image $I$, the normal map $N$, and the estimated albedo $A$ as input and outputs an albedo residual $\delta\bar{A}$,

$$\delta\bar{A} = f_{\bar{A}}(I, N, A). \tag{6.9}$$

The refined albedo, $\bar{A} = A + \delta\bar{A}$, then replaces the original albedo $A$ as the input to the rest of the model ($f_S$ and $f_{\delta R}$) to predict a new relit output $\bar{R}$. As shown in Figure 6.11, the refined albedo after adaptation can recover input details (cloth color in this case) that are not seen in synthetic data.

When finetuning on real data, we only train $f_{\bar{A}}$ while keeping the rest of the model fixed. The loss functions include a similarity loss, an identity loss, a GAN loss, and two lighting

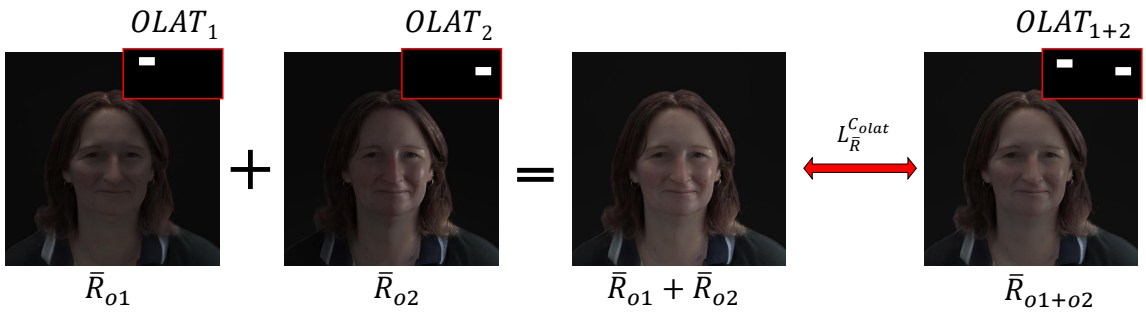



**Figure 6.12.** OLAT consistency loss. We relight the input using two randomly picked OLAT maps and compute the difference between the sum of them and the result when we relight using the sum of the two OLAT maps.

consistency losses. For all losses, the superscript refers to the loss type, while the subscript refers to the output where the loss is computed on.

**Lighting consistency loss $\mathscr{L}_{\bar{R}}^{C_{olat}}$, $\mathscr{L}_{\bar{R}}^{C_{rela}}$**

To keep the relighting effect consistent before and after the syn2real adaptation, we apply the following consistency losses between the relit image $\bar{R}$ and the input foreground image $I$ as regularization terms.

First, we use the *OLAT lighting consistency loss*. This loss assumes that the illuminations and relit outputs have the distributive property, i.e., summing two environment maps and relighting with the sum leads to the same result as summing the two relit outputs, as shown in Figure 6.12. In particular, we first generate a set of OLAT environment maps, where only one pixel is lightened in each of them to simulate the light stage scenario. We demonstrate the OLAT images we used to train and evaluate our network in Figure 6.13. In total, we uniformly sample 168 locations on a sphere. We put a high-intensity light source at each location and project them to latitude-longitude format to create 168 OLAT HDR environment maps in total. Each light source in our OLAT HDR environment is created such that the light source preserves the same intensity when it is mapped onto the sphere from the latitude-longitude representation.

During each training iteration, we randomly sample two of them ($OLAT_1$ and $OLAT_2$) to simulate two different light source locations and get the combined environment map $OLAT_{1+2}$, which simulates two light sources turned on at the same time. Let the output images relit by

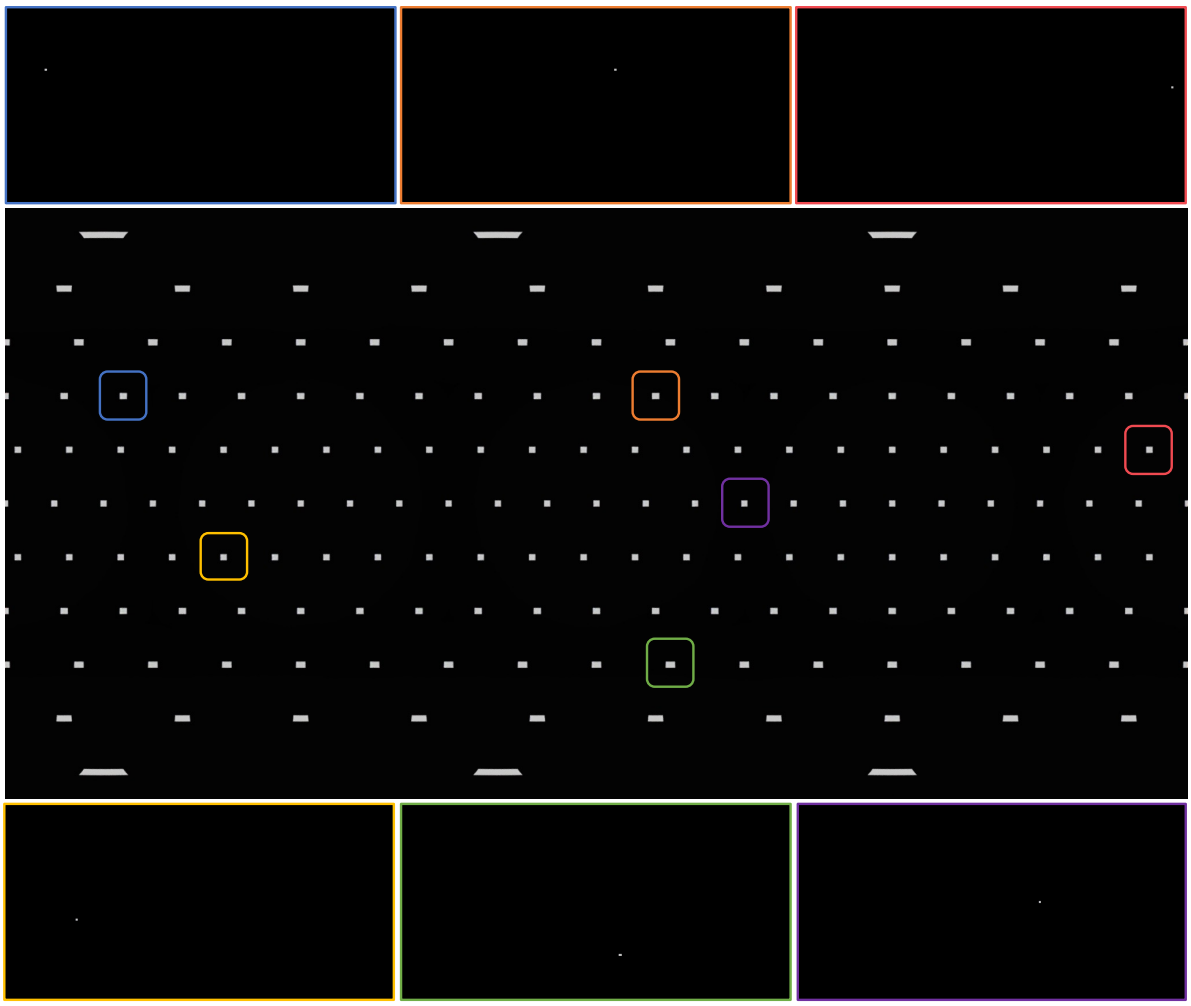


**Figure 6.13.** We created OLAT-like HDR environment maps at uniformly sampled locations on a sphere. In the middle we show all the sampled locations (all lights on). In the top and bottom row, we show different OLAT environment maps (one light on) at 6 locations.

maps $OLAT_1$, $OLAT_2$, and $OLAT_{1+2}$ be $\bar{R}_{o1}$, $\bar{R}_{o2}$, and $\bar{R}_{o1+o2}$, respectively. The loss is then

$$\mathscr{L}_{\bar{R}}^{C_{olat}} = ||(\bar{R}_{o1} + \bar{R}_{o2}) - \bar{R}_{o1+o2}||_1. \tag{6.10}$$

Using two OLAT images, instead of two environment maps, ensures that the relighting can react to high-frequency point lighting conditions such as the OLAT lighting as shown in the accompanying video.

Second, we use the *relative lighting consistency loss*. This loss assumes that the adaptation should only change the photorealism and acts as a form of regularization to preserve the

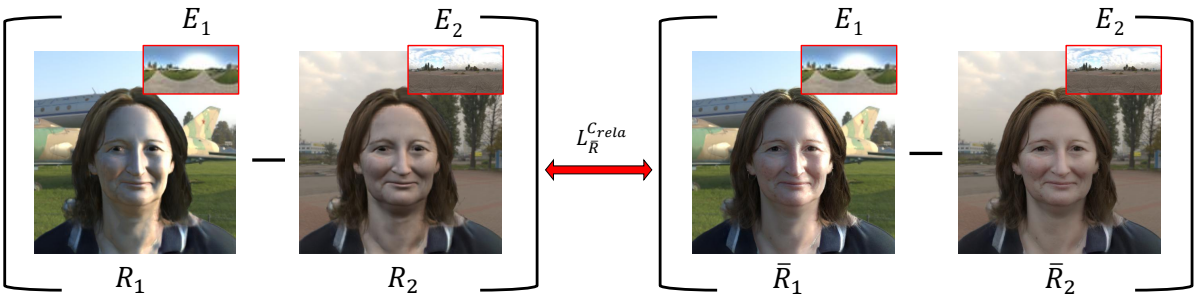



**Figure 6.14.** Relative consistency loss. We relight the input using two randomly picked environment maps and compute the difference between the two relit outputs. This is done for both before and after the syn2real adaptation, and we compute the loss between the two differences.

relighting behaviors that our network has learned from a physically-based renderer. As we do not have ground truth lighting, we enforce this idea through a pair of relighting outputs. Let $E_i$ and $E_j$ be an arbitrary pair of two environment maps randomly picked every time the loss is evaluated. We denote the relit results using $E_i$ and $E_j$ as $R_i$, $R_j$ before adaptation and $\bar{R}_i$, $\bar{R}_j$ after adaptation. We then apply an $L_1$ loss between the two differences,

$$\mathscr{L}_{\bar{R}}^{C_{rela}} = ||(R_i - R_j) - (\bar{R}_i - \bar{R}_j)||_1. \tag{6.11}$$

This way, the network has a stronger incentive to preserve the lighting produced before the syn2real adaptation, as the deviations from the original outputs will be discouraged. Mathematically, $||(R_i - R_j) - (\bar{R}_i - \bar{R}_j)||_1 = ||(R_i - \bar{R}_i) - (R_j - \bar{R}_j)||_1 = ||d_i - d_j||_1$. This encourages the improvements $d$ to be invariant to lighting conditions. It means that our loss will keep the overall lighting consistent before and after the adaptation but only change the photorealism. We visualize the effect of the relative lighting consistency loss in Figure 6.14 and Figure 6.15. We note that the relative consistency loss plays an important role in our adaptation framework. Figure 6.15 compares three variants: without the adaptation (*w/o adaptation*), without relative lighting consistency loss (*w/o relative consistency loss*), and our full model. As can be seen, *w/o adaptation* (first row) leads to washed out albedo details and textures (e.g., on clothing). With adaptation, but without the relative lighting consistency (second row), the method fails to recover

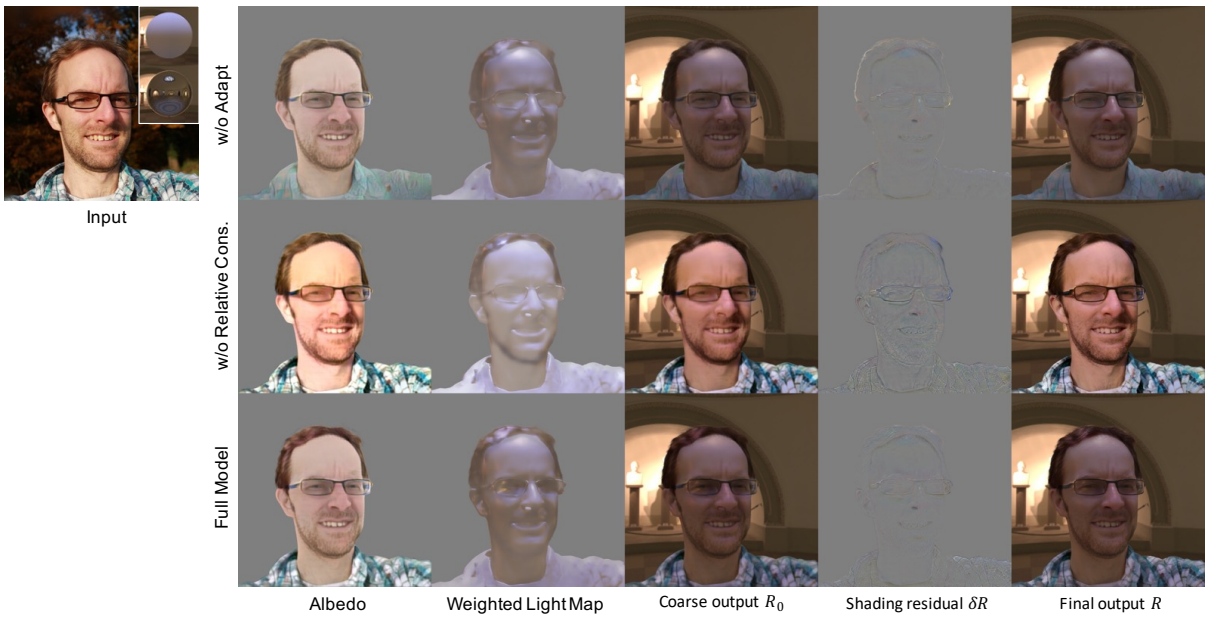



**Figure 6.15.** Comparisons on the effectiveness of the albedo adaptation and relative lighting consistency loss for the output layers. Without adaptation, the model cannot keep facial details such as hair colors and beards, and most notably the textures on the clothing. Without the relative lighting consistency loss, the input image lighting is baked into the predicted albedo and leads to inaccurate relighting results. Our full model can recover correct albedo colors and details better, since the proposed *relative lighting consistency loss* aims to improve the photorealism in isolation under arbitrary target lightings.

the correct albedo. For example, the cast sunlight in the input is baked into the predicted albedo and the model is unable to disentangle the lighting and the albedo in the input. With our full model (third row), both the details and correct albedo colors can be recovered after adaptation.

**Similarity loss** $\mathscr{L}_{\bar{A}}^{sim}$

To prevent the refined albedo from deviating too much from the original albedo, we extract VGG features from the original and refined albedo and put an $L_1$ loss between them.

**Identity loss** $\mathscr{L}_{\bar{R}}^{Id}$

To ensure the identity does not get changed after relighting, we apply a face recognition network [163] on the input $I$ and relit images $\bar{R}$, extract high-level features from them, and compute the $L_1$ distance between these two features.

**GAN loss $\mathscr{L}_{\bar{R}}^{G}$**

To make the relit images look realistic, we also put a GAN loss on the relit images $\bar{R}$.

The overall loss is then

$$\begin{aligned}\mathscr{L}^{adap} =& \lambda^{C_{olat}} \mathscr{L}_{\bar{R}}^{C_{olat}} + \lambda^{C_{rela}} \mathscr{L}_{\bar{R}}^{C_{rela}} \\ &+ \lambda^{sim} \mathscr{L}_{\bar{A}}^{sim} + \lambda^{Id} \mathscr{L}_{\bar{R}}^{Id} + \lambda^{G} \mathscr{L}_{\bar{R}}^{G}\end{aligned} \tag{6.12}$$

where $\lambda$'s are the weights and are set to 10, 10, 10, 1, 1 respectively in our framework.

Example results before and after the domain adaptation can be found in Figure 6.21. We also show comparisons with directly applying syn2real on the output relit image as well as without adopting our improvement in Section 6.4.2. The losses are the same as the full model, while the similarity loss is computed on the relit image instead of the albedo. We found that both alternative approaches give the network too much "freedom" to change the relit output. As a result, the correct lighting information is not preserved after adaptation. In particular, when directly applying syn2real to the output, it creates a "shortcut" that the network just mimics the real image since the real image is both the input to the network and the target. On the other hand, when we do not adopt the modification in Section 6.4.2 and directly estimate the output using a black box $f_R$, the network tends to use the albedo map in an unpredictable way to generate the relit image. In our approach, the generation is fully controllable by the albedo map since a linear combination is used.

### 6.4.4 Relighting Portrait Videos

In many applications, portrait video relighting is required. A straightforward way to achieve this capability is to simply apply the relighting model in a frame-by-frame manner. However, this usually results in flickering artifacts in the relit video. In particular, we observe that the flickering is due to the intermediate outputs in our framework changing too frequently between frames. These intermediate outputs include the estimated normal and albedo maps.

To resolve the issue, we learn two temporal residual networks $f_{\widetilde{N}}$ and $f_{\widetilde{A}}$ to make the normal and albedo maps temporally consistent, respectively. Note that these temporal networks are only required for video portrait relighting and are learned while keeping the other parts of the framework fixed.

The temporal residual networks take outputs from the past frame and the predictions from the image-only relighting part of the framework at the current frame as inputs. In particular, $f_{\widetilde{N}}$ takes the original normal prediction $N_t$ at time $t$ and the refined normal from the previous frame $\widetilde{N}_{t-1}$ at time $t-1$ as input for predicting a temporal normal residual $\delta\widetilde{N}_t$. This residual is added to the original normal prediction $N_t$ to obtain the refined normal $\widetilde{N}_t$.

$$\delta\widetilde{N}_t = f_{\widetilde{N}}(\widetilde{N}_{t-1}, N_t), \quad \widetilde{N}_t = N_t + \delta\widetilde{N}_t. \tag{6.13}$$

Similarly, $f_{\widetilde{A}}$ takes temporally refined normal $\widetilde{N}_{t-1}$, $\widetilde{N}_t$, temporally refined albedo $\widetilde{A}_{t-1}$, and the albedo $\bar{A}_t$ (after syn2real adaptation) as input, and predicts a temporal albedo residual $\delta\widetilde{A}_t$. This residual is added to the syn2real albedo $\bar{A}_t$ to obtain the temporally refined albedo $\widetilde{A}_t$.

$$\delta\widetilde{A}_t = f_{\widetilde{A}}(\widetilde{N}_{t-1}, \widetilde{N}_t, \widetilde{A}_{t-1}, \bar{A}_t), \quad \widetilde{A}_t = \bar{A}_t + \delta\widetilde{A}_t. \tag{6.14}$$

After we obtain the refined normal and albedo maps, they are fed into the rest of the framework to obtain the final relit output $\widetilde{R}_t$. To train the two networks, we again adopted the similarity loss and identity loss used in Section 6.4.3. To ensure temporal consistency, we also add a warping loss between neighboring frames.

**Similarity loss $\mathscr{L}^{sim}_{\widetilde{N}}$, $\mathscr{L}^{sim}_{\widetilde{A}}$**

Similar to our syn2real treatment in Section 6.4.3, we add a perceptual loss between the original ($N_t$ and $\bar{A}_t$) and temporally-refined attributes ($\widetilde{N}_t$ and $\widetilde{A}_t$) to prevent the refined attributes from deviating too much from the original ones.

**Identity loss $\mathscr{L}^{Id}_{\widetilde{R}}$**

To ensure the identity does not get changed after refinement, we extract identity features from input $I_t$ and refined output $\widetilde{R}_t$ using a face recognition network [163] and put an $L_1$ loss between them.

**Warping loss $\mathscr{L}^{W}_{\widetilde{R}}$**

We first compute the optical flow [182] on neighboring input frames $I_{t-1}$ and $I_t$. We then apply the flow to warp the previous relit frame $\widetilde{R}_{t-1}$ and compute its difference to the current relit frame $\widetilde{R}_t$,

$$\mathscr{L}^{W}_{\widetilde{R}} = \|\mathscr{W}(\widetilde{R}_{t-1}) - \widetilde{R}_t\|_1, \tag{6.15}$$

where $\mathscr{W}$ is the warping function based on the estimated optical flow. The final loss is then

$$\mathscr{L}^{tem} = \lambda^{sim}\left(\mathscr{L}^{sim}_{\widetilde{N}} + \mathscr{L}^{sim}_{\widetilde{A}}\right) + \lambda^{Id}\mathscr{L}^{Id}_{\widetilde{R}} + \lambda^{W}\mathscr{L}^{W}_{\widetilde{R}}, \tag{6.16}$$

where $\lambda$'s are loss weights and are set to $0.2, 0.1, 5$ in our framework. In Figure 6.23, we show that our temporal refinement can improve temporal consistency significantly.

### 6.4.5 Datasets and Implementation Details

For supervised learning of our portrait relighting network (Section 6.4.2), we use our synthetic dataset created from Section 6.3 for training. For syn2real finetuning on real data (Section 6.4.3), we crop the original FFHQ [98] data with a similar field of view to our synthetic data and resize the images to 512$x$512 resolution. The FFHQ dataset contains 70k images, which are split into 60k/10k for training/validation. We use the training set in our adaptation and leave the validation set for evaluation. For temporal refinement (Section 6.4.4), we use the TalkingHead-1kH dataset [231], which consists of about 500k face video clips. We randomly sample 4 consecutive frames from a clip during each iteration to train the temporal network.

For both our syn2real and temporal networks ($f_{\widetilde{A}}, f_{\widetilde{A}}, f_{\widetilde{N}}$), we use a ResNet with 2

downsample layers, 9 residual blocks, and 2 upsampling layers. The channel size for the first layer is 32 and is doubled after each downsampling. We train our networks using a learning rate of 0.0001. For each of the three stages (synthetic, syn2real, and temporal), we train for 50k iterations using a batch size of 64 on 8 NVIDIA V100 GPUs. All of our networks are trained to produce an image of 512 resolution. Our end-to-end process runs in 65ms on an NVIDIA A6000 GPU.

## 6.5 Experiments

We compare our method with SIPR-W [234], TR [160], and NVPR [263]. Since none of these methods releases their code or models, we requested the authors to apply their models to our inputs and provide the results to us. Note that for NVPR, the authors did not return the results for the synthetic test set, so we only compare with NVPR on the real dataset. In the following, we first show comparisons on our synthetic test set (Section 6.5.1). We then show comparisons on a real in-the-wild face dataset using FFHQ (Section 6.5.2). Next, we perform ablation studies on our network design choices (Section 6.5.3). Finally, we show additional features available in our framework, including controlling glasses glares and relighting portrait videos with temporal consistency (Section 6.5.4).

### 6.5.1 Comparisons on the Synthetic Dataset

Our synthetic test set consists of 25 identities that are never seen during training. We use all the available expressions for each identity and randomly pair each of them with two environment maps to form a ground-truth pair. This results in 477 pairs of images in total.

#### Quantitative results

We compare the mean absolute error (MAE), the mean squared error (MSE), the structural similarity index measure (SSIM) [235], and the Learned Perceptual Image Patch Similarity (LPIPS) [266] between the relit images and the ground truths for each method. The comparison


**Figure 6.16.** Comparisons with state-of-the-art methods on our synthetic dataset. SIPR-W [234] and TR [160] tends to change skin tone and hair color. Ours can predict the relit results closer to the ground truth and preserve the identities, skin tone, hair color, and the facial details.

**Table 6.3.** Quantitative evaluations on the test set of our synthetic dataset.

| | MAE ↓ | MSE ↓ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|---|
| SIPR-W [234] | 0.2506 | 0.1159 | 0.4406 | 0.3633 |
| TR [160] | 0.2328 | 0.1083 | 0.4851 | 0.3465 |
| Ours | **0.1967** | **0.0975** | **0.5272** | **0.2891** |

is shown in Table 6.3. Note that all the baselines and our method suffer from the similar domain gap, i.e., trained on real data but evaluated on synthetic data. Our method achieves the best results on all metrics, outperforming all other baselines by a large margin.

**Qualitative results**

The qualitative comparisons are shown in Figure 6.16. Compared with other methods, the results generated by our approach are more similar to the ground truth in terms of both lighting accuracy and color tones. The results by SIPR-W have multiple artifacts, especially on the hair and clothes. The results by TR have fewer artifacts but sometimes change the person's identity or skin colors. This may be expected since TR is never trained on synthetic data. Hence, we compare the performance of the challenging in-the-wild images next.

### 6.5.2 Comparisons on the Real Dataset

We compare the performance of the competing methods on real images in the FFHQ dataset. We pick 116 images from the FFHQ test set and randomly pair each of them with 4 different environment maps, resulting in 464 images in total. The images are chosen with gender, age, and ethnic diversities in mind.

**Qualitative results**

In Figure 6.17, we provide qualitative comparisons in a tighter crop, accommodating the preferred setting in SIPR-W and NVPR. SIPR-W has trouble handling the real-world data since the model is only trained with synthetic data. NVPR tends to predict unnatural illumination on the face regions and incorrect appearances on clothes regions. TR is able to generate photorealistic


**Figure 6.17.** Comparisons with state-of-the-art methods on FFHQ evaluation dataset for real portraits. SIPR-W [234] cannot handle in-the-wild images due to the models purely trained with synthetic data. NVPR [263] generates unnatural appearance and artifacts on cloth regions. TR [160] can generate plausible relit images, but exhibits artifacts on hair (second and fourth rows), inconsistent lighting directions with the reference diffuse sphere (third row), and lack of facial details (sixth row). Our results can synthesize plausible glares on eyeglasses (first row), keep facial and hair details, and generate relit portraits consistent with the target illumination.


**Figure 6.18.** Comparisons with state-of-the-art methods on FFHQ evaluation dataset for *larger* (including shoulder) real portraits. TR [160] tends to generate blurred details for hair regions and shows artifacts on clothes. TR also exhibits the relit face that is inconsistent with the given lighting, compared to the diffuse sphere reference (third column). Ours can remove unwanted cast shadows from eyeglasses (first column), keep details and colors on clothes and hair, and overall consistent lighting with the diffuse sphere reference.

results but exhibits artifacts on hair regions (second and fourth rows) and loses facial details (sixth row). In addition, the illumination by TR is sometimes not aligned with the target. In the third row, the ground snow is supposed to illuminate the face from the bottom as shown in the diffuse sphere reference, while the TR output illuminates the face from the right side. Our method generates a natural appearance and consistent lighting conditions with the reference sphere and additionally adds glares on eyeglasses (first row). In Figure 6.18, we compare our results with the TR method in larger crops. While TR can overall generate photorealistic results, it shows unnatural appearance and artifacts in the hair regions and an inconsistent lighting output (third column). It also shows blurry results in the hair region (fourth column). Our method can robustly handle the wide variety of hair colors and clothes types thanks to the variations in our synthetic data and in-the-wild data used in the synthetic-to-real adaptation.

**Table 6.4.** Quantitative evaluations on the test set of FFHQ.

| | Lighting ↓ | FID ↓ | Identity ↑ |
|---|---|---|---|
| SIPR-W [234] | - | 73.31 | 0.6522 |
| NVPR [263] | - | 52.25 | **0.7461** |
| TR [160] | 0.0699 | 43.50 | 0.6226 |
| Ours | **0.0645** | **36.13** | 0.7413 |

### Quantitative results

Since we do not have ground truths for quantitative comparisons, we adopt the following metrics. We measure the resulting quality of each method in three different aspects: lighting accuracy, photorealism, and identity preservation.

### Lighting recovery

To evaluate how accurately the relit results translate the lighting information from the input lighting environment, we estimate the lighting from relit images and compare it with the target lighting. In particular, we use the relit portrait, along with the estimated albedo and normal maps, to compute the second-order spherical harmonic (SH) coefficients [180] by solving a least square problem on visible skin pixels. We use face *skin* and *nose* masks predicted from a PyTorch implementation of a face parsing network[8] based on BiSeNet [259] to determine skin pixels. This is done only for TR [160] and our method since both methods generate albedo and normal maps as side products. We reconstruct a sphere rendering from estimated SH coefficients and apply a global intensity multiplier to compensate for a global exposure difference between the estimated and target illumination. To be specific, we compute the average intensity of the sphere rendering from the estimated SH coefficients to derive the global scalar for matching the average intensity of the sphere rendering from the target SH coefficients. We compute the L1 pixel difference between the reconstructed and target sphere renderings to measure the accuracy. We computed this error in the pixel space instead of the coefficient space since the coefficient space has ambiguity (i.e., multiple coefficient combinations can give rise to the same reconstruction). The comparisons

[8]https://github.com/zllrunning/face-parsing.PyTorch

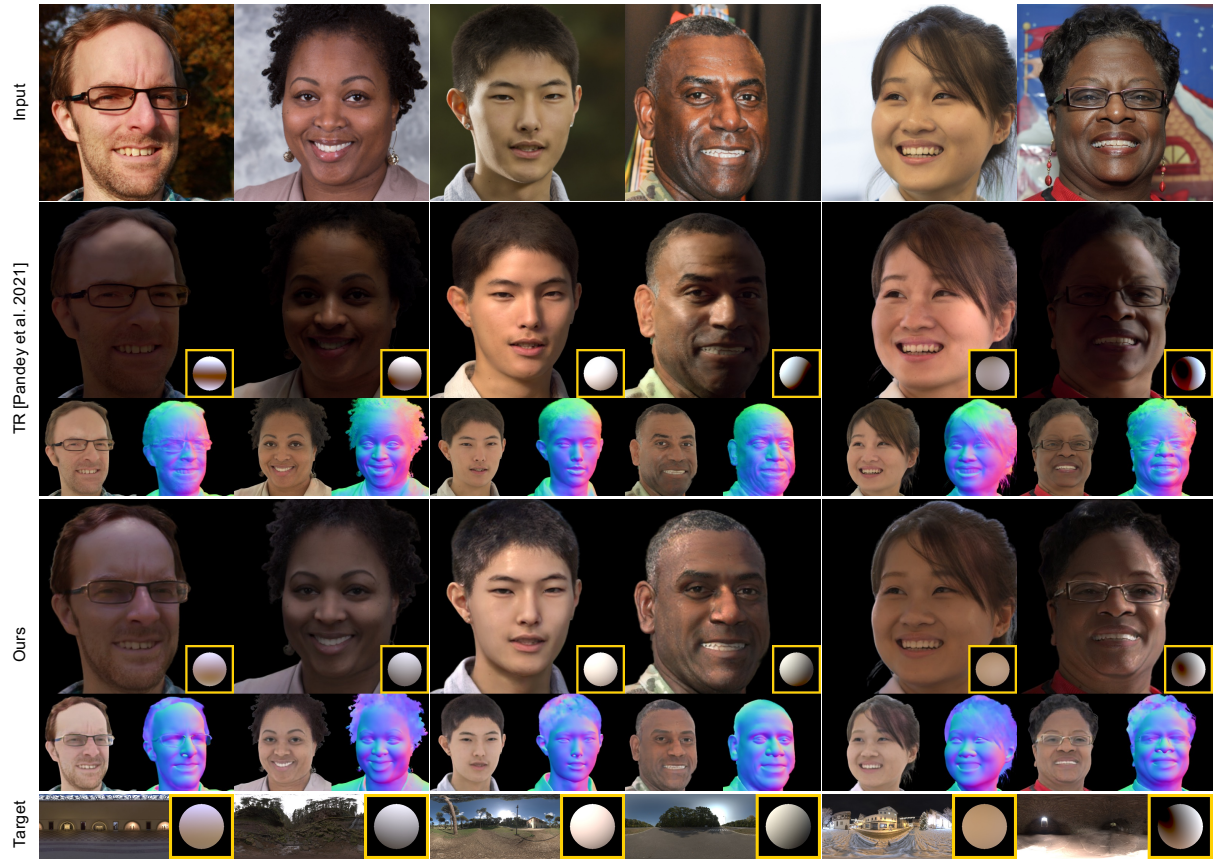



**Figure 6.19.** Comparison of lighting recovery from relit images on the FFHQ dataset. We compare TR (second row) and our method (third row) on the consistency of the lighting information recovered from the relit portrait via the second-order spherical harmonic (SH) lighting [180] compared to the ground truth environment (bottom row). Given the relit image (top of the row), albedo (bottom left of the row), and normal map (bottom right of the row) generated from the input image (top row), we estimate the second-order SH coefficients by solving a linear system using foreground pixels. We visualize the recovered SH lighting condition on the diffuse ball rendering in the inset highlighted in yellow. The inset diffuse ball renderings show our method has more consistent lighting than TR compared to the ground truth diffuse ball rendering (bottom row).

are shown in Figure 6.19 and Table 6.4. Our results demonstrate better numerical accuracy and are closer to the ground truth sphere rendering. Although the intermediate normal predictions from TR might contain more high-frequency details than ours, it sometimes predicts incorrect geometry, such as the eyeglasses shadow in the cheek region (first column in Figure 6.19). When the entire framework is trained end-to-end, the final rendering network is able to compensate for small differences or errors from the normal prediction.

**Synthesis quality**

To compare the photorealism of the resulting quality, we compute Fréchet Inception Distance (FID) [76, 194] scores between the input FFHQ test set and the relit results from each method. For fair comparisons, we mask out the background of each image and only consider the foreground region. From Table 6.4, it can be seen that our method has the lowest FID score, which demonstrates its ability to synthesize realistic portrait images.

**Identity similarity**

We evaluate the identity consistency before and after relighting, as shown in Table 6.4. To do so, we use a face recognition network [7] (different from the one that we use in our training loss) to extract face embeddings before and after relighting of foreground images for each method and compute the similarity score between these two embeddings. We found that NVPR achieves a slightly higher identity similarity score, but our score is very close to theirs, and our FID score is much better. Moreover, from Figure 6.17 it can be seen that their method shows unnatural facial appearance that is not consistent with the target illumination (fourth and sixth rows).

**User study**

Finally, we perform a user study by asking Amazon Mechanical Turk workers to answer three binary questions: given an input image and an environment map, which relit image *1) has more consistent lighting with the environment map? 2) maintains better facial details? 3) keeps the original identity better?* between our relighting result and one of the baselines. The orders

**Table 6.5.** User study on the FFHQ test set. For each comparison, we ask the workers to choose the better image according to the criterion on the left. The preference rates where workers prefer our image than the other baselines are summarized below. For all metrics, the preference rates are greater than 0.5, meaning the workers prefer our results more.

| | Ours vs. SIPR-W | Ours vs. NVPR | Ours vs. TR |
|---|---|---|---|
| Consistent Lighting | 0.5453 | 0.5934 | 0.5474 |
| Facial Details | 0.8527 | 0.7601 | 0.5409 |
| Similar Identity | 0.8886 | 0.8218 | 0.5230 |

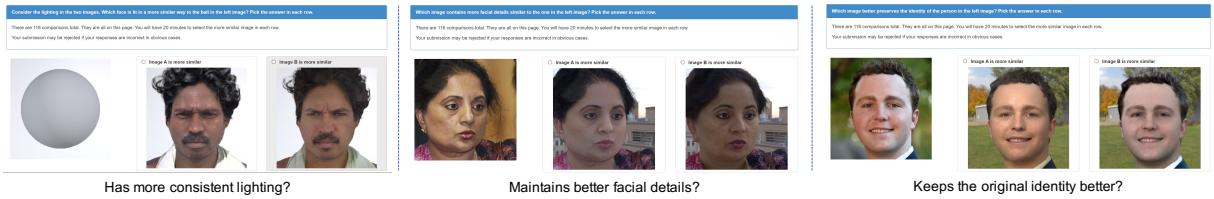



**Figure 6.20.** Examples of user study questions interface for the three questions: *1) has more consistent lighting? 2) maintains better facial details? 3) keeps the original identity better?* For the first question, we present a diffuse ball lit by the target environment map and ask users to select an option from two relit images, which is lit in a more similar way to the ball. For the last two questions, we present an input portrait image and ask users to select an option on relit results from two methods.

of the methods are randomized for fair comparisons. To make the workers better understand the concept of consistent lighting, we also show the relit balls similar to the ones shown in Figure 6.17. We split the 464 images into 4 partitions and ask each worker to evaluate a partition on a single question. Each comparison is evaluated by 3 different workers, resulting in 1392 comparisons for each question. In total, 4 (partitions) x 3 (questions) x 3 (comparisons) = 36 workers participated who have never seen the questions. The evaluation results are shown in Table 6.5. It can be seen that workers prefer our results more compared to all baselines using all different metrics.

We demonstrate our user study interface for the questions: given an input image and an environment map, which relit image *1) keeps the original identity better? 2) maintains better facial details? 3) has more consistent lighting with the environment map?* between our relighting result and one of the baselines in Figure 6.20. For the first two questions, we present an input portrait image and ask users to select an option on relit results from two methods. For the last

**Table 6.6.** Ablation study on baseline adaptation methods with real FFHQ test set. All the results are computed on foreground images. There are trade-offs between identity consistency, synthesis quality (FID) and OLAT consistency. The model can report a slightly higher FID score or the identity similarity when it is trained *without* relative or OLAT consistency loss at the cost of the much worse lighting consistency, which is less suitable for portrait relighting. Our full model achieves the **best** performance in OLAT consistency while maintaining the similar scores (second best) in the other metrics, which leads to the best lighting control while preserving the portrait identities with high photorealism.

| | OLAT Cons. ↓ | FID ↓ | Identity ↑ |
|---|---|---|---|
| w/o improved render net | 0.0970 | 54.08 | 0.7237 |
| w/o adaptation | 0.0417 | 58.61 | 0.6402 |
| Finetune only | 0.2284 | 69.24 | 0.6932 |
| Direct syn2real | 0.0267 | 43.14 | 0.7206 |
| w/o Relative consistency loss | 0.0746 | 36.90 | **0.7612** |
| w/o OLAT consistency loss | 0.0205 | **35.87** | 0.7405 |
| Our full model | **0.0166** | 36.13 | 0.7413 |

question, we present a diffuse ball that is lit by the target environment map and ask users to select an option from two relit images which is lit in a more similar way to the ball.

### 6.5.3 Ablation Studies

We compare different baselines for our relighting model design choices described in Section 6.4.2 and Section 6.4.3. We compute the same metrics as used in Section 6.5.1 for synthetic test set and in Section 6.5.2 for the real test set in Table 6.6. In addition, we also compute the OLAT consistency error to evaluate the lighting control of our relighting performance. The computing procedure is similar to our OLAT lighting consistency loss (Equation 6.10). The qualitative results of each ablative baseline are shown in Figure 6.21 on the FFHQ real test set. We describe each ablative baseline as follows:

**w/o improved render network**

Instead of using the proposed rendering decomposition and delta relit image prediction, we directly use a blackbox rendering network in TR [160] to predict the relit images.


**Figure 6.21.** Ablation comparisons on our adaptation choices. For *w/o adaptation*, the model cannot perform well on real data and cannot keep the facial details such as hair color and beards. For *w/o Improved Render Net*, *Finetune Only*, and *Direct Syn2Real*, they give too much freedom to the network which also leads to degenerated results due to overfitting, including out-of-control relighting results and losing facial or clothing details. With our proposed adaptation method, *w/o Relative consistency loss* cannot synthesize minor facial highlights and might change hair or face colors. For *w/o OLAT consistency loss*, the overall results are similar to our full model, but the lighting control is slightly worse and the output exhibits artifacts on the hairlines (first, second and fourth rows).

**w/o adaptation**

We only use the proposed improved rendering network (Section 6.4.2) without applying the syn2real adaptation (Section 6.4.3).

**Finetune only**

We apply all the syn2real adaptation losses mentioned in Section 6.4.3 to finetune all the networks of our base model (Section 6.4.2) after training with synthetic data.

**Direct syn2real on relit image**

Instead of learning an albedo residual $\delta\tilde{A}$ as mentioned in Section 6.4.3, we learn a module to predict the residual for the output relit image while keeping the weights for the other modules in the base model fixed.

**w/o relative consistency loss**

We keep the weights of our base model unchanged and only optimize the weights for the albedo residual network with all losses for adaptation except the relative consistency loss.

**w/o OLAT consistency loss**

Similar to above, we only optimize for the albedo residual network with all adaptation losses except the OLAT consistency loss.

**Full adaptation**

Our full adaptation method optimizes the albedo residual network with all the adaptation losses.

As shown in Figure 6.21, *w/o adaptation* cannot generalize to real portrait images well. It predicts incorrect appearances on hair and clothing regions. For *w/o improved render net*, *Finetune only*, and *Direct syn2real*, they give too much freedom to the adaption network during training, which leads to incorrect relighting results due to overfitting. In particular, the overall appearances do not reflect the input environment maps for *Finetune only*. The facial and clothing details are not preserved well in *w/o improved render net*. For *Direct syn2real*, although the relighting results look plausible, it loses the details of clothing regions. On the other hand, *w/o relative consistency loss* loses minor relighting details such as the specular highlights on the skin (third row) and might also change hair or face colors. For *w/o OLAT consistency loss*, the relit image exhibits artifacts around the hairline (first, second and fourth columns).

We also would like to note there are trade-offs between identity similarity, synthesis quality (FID), and lighting control. One possible way for the model to optimize for the identity similarity and the FID score is to try not to change image content as much as possible at the cost of violating lighting consistency. As seen in Table 6.6, *w/o OLAT consistency loss* and *w/o relative consistency loss* achieve the slightly better FID and identity scores but also show much worse lighting consistency. This is reinforced in Figure 6.21 that without the lighting consistency losses, the outputs exhibit unnatural results and artifacts less suitable for relighting.

Input w/o Glasses Glares w/ Glasses Glares

**Figure 6.22.** Example comparisons showing results without vs. with adding glares using our method. The diffuse and mirror balls are shown in the set. ***Please click each row to view the video.***

We choose our full model with the **best** performance on OLAT consistency while keeping the identity similarity and FID scores as the second best. This leads to the best lighting control on the relit images with the least artifacts while largely preserving the input identities with high photorealism.

### 6.5.4 Video Relighting

Finally, we show results using our video relighting framework.

**Table 6.7.** Quantitative comparisons for temporal consistency on relit videos. For each relit video, we use optical flows to warp neighboring frames and compute the differences.

| | MAE ↓ | MSE ↓ |
|---|---|---|
| Ours (w/o temporal) | 0.0511 | 0.0024 |
| NVPR | 0.0488 | 0.0031 |
| Ours (w/ temporal) | **0.0477** | **0.0020** |

**Controllable glasses glares**

Thanks to our synthetic dataset and our new network architecture (Section 6.4.2), we are able to control the glares on glasses in the relighting results. This is achieved by three of our design choices. First, we have both normals with and without lenses in our synthetic dataset, so we are able to supervise our model on both normals. Second, we modify our geometry network and specular network to consider normals with lenses, including an additional light map with a very high Phong exponent. Finally, we directly use the light maps to first generate a coarse relit image and predict the residual for the output (Equation 6.5) instead of using a neural network as a black box. Therefore, differences in light maps are directly transferable to the final output.

Example results are shown in Figure 6.22 in the embedded videos. Note that the glares are caused by the reflections from the rotated environment map.

**Temporally-consistent video relighting**

We are able to generate temporally-consistent outputs by refining the intermediate normal and albedo maps based on previous frames (Section 6.4.4). An example comparison is shown in Figure 6.23 in the embedded video. Without the temporal refining components, the results exhibit flickering artifacts when the subject is making large motions like rotating the head abruptly. When utilizing our temporal refining networks, the results become much smoother and more realistic.

We also compare our video relighting results with NVPR [263] in Figure 6.24 and Table 6.7. We evaluate temporal consistency on 9 different subjects with 6 selected environment maps (resulting in 54 videos). We compute optical flows [182] on neighboring frames and

Input w/o Temporal w/ Temporal

**Figure 6.23.** Comparisons between with and without applying our temporal refinement. ***Please click each row to view the video.***

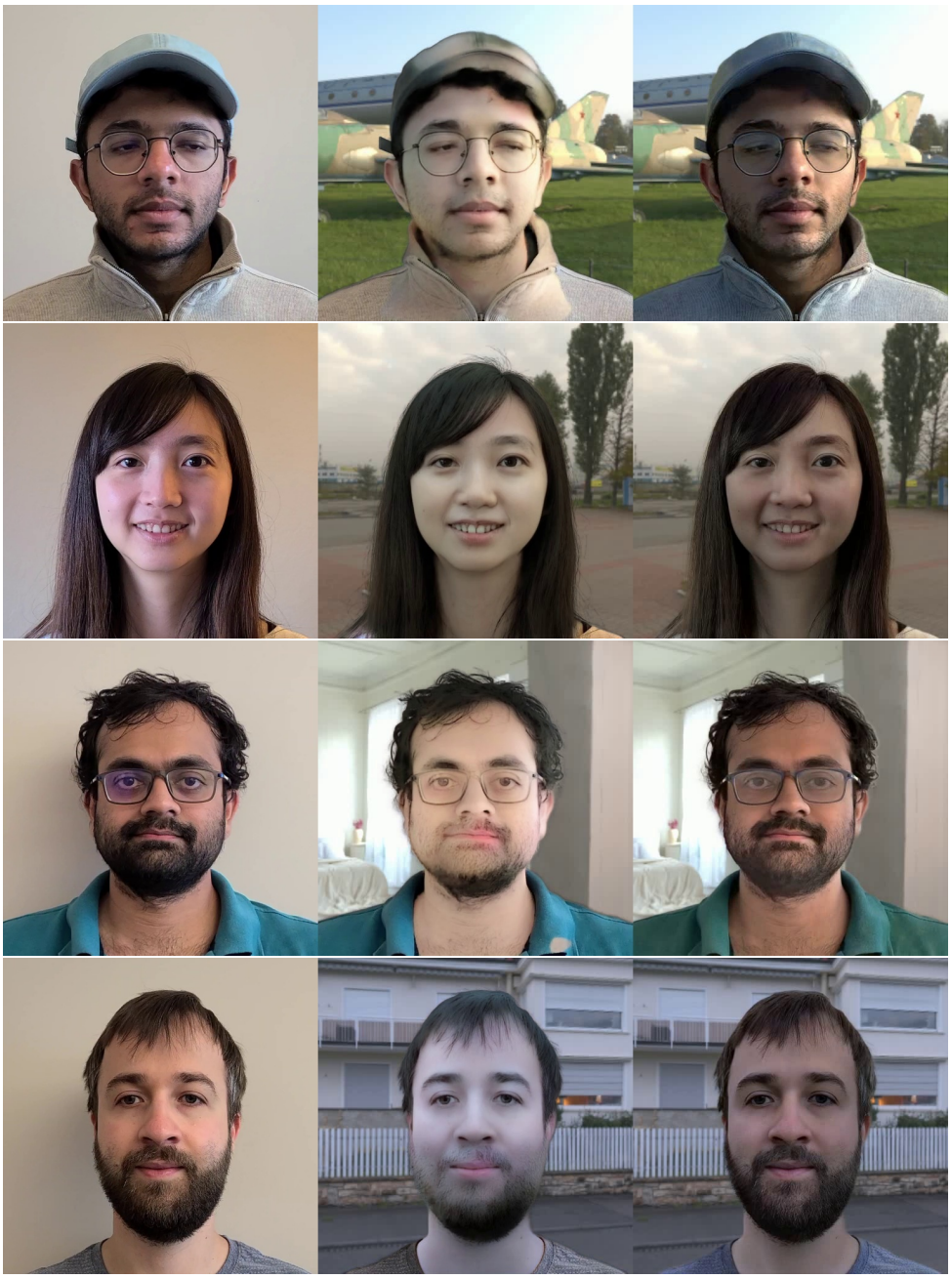
warp the frame from $t$ to $t+1$ and compute the MAE and MSE between the warped frames and estimated frames. A lower error means the video contains fewer flickering artifacts. Ours performs better than NVPR in terms of temporal consistency and relighting quality.

### 6.5.5 Limitations

We demonstrate a few failure cases in Figure 6.25. Our method cannot predict the correct albedo for the overly exposed highlights on hair under a strong light source. It also cannot remove the reflection on the eyeglasses and the shadows cast by a direct light on the clothes. The inaccurate albedo predictions lead to incorrect relighting results. We could ameliorate this effect by first using an off-the-shelf method to remove these unwanted highlights or shadows and then applying our relighting model. For future work, we could explicitly model these strong shadows [270] or reflections to learn better albedo and relighting predictions.

## 6.6 Conclusion

We proposed a new synthetic dataset for single image portrait relighting, which contains over 500 subjects of diverse ages, genders, and races with randomly assigned hairstyles, clothes, and accessories, rendered using a physically-based renderer. We showed that the single image and video portrait relighting tasks can be tackled without light stage data by training a base relighting model on the diverse synthetic dataset and later applying a synthetic-to-real domain adaptation



Input NVPR [263] Ours

**Figure 6.24.** Comparisons with the state-of-the-art method on video relighting. Note our results exhibit fewer flickering artifacts and are more temporally consistent. ***Please click each row to view the video.***


**Figure 6.25.** Examples of some failure cases. Our albedo estimation cannot remove strong highlights on eyeglasses and hair. The strong shadow cast on the clothes from the back of the subject is difficult to remove. The inaccurate albedo estimations lead to incorrect relit outputs.

using real in-the-wild portrait images and video. Self-supervised lighting consistency losses are used to ensure lighting consistency during the adaptation learning. Extensive experimental validations suggested that our relighting method outperforms the latest state-of-the-art single image or video portrait relighting methods in terms of lighting recovery, synthesis quality, and identity preservation. We believe our synthetic dataset for relighting and our method that does not require high-quality captured data fosters the portrait relighting research.

Chapter 6 is based on the material as it appears in ACM Transactions on Graphics, 2022 (”Learning to Relight Portrait Images via a Virtual Light Stage and Synthetic-to-Real Adaptation”, Yu-Ying Yeh, Koki Nagano, Sameh Khamis, Jan Kautz, Ming-Yu Liu, Ting-Chun Wang ). The dissertation author was the primary investigator and author of this paper.

# Chapter 7

# Conclusion and Future Work

This dissertation explores various approaches to creating photorealistic content for diverse subjects, including microfacet surfaces, transparent objects, and portraits, utilizing both optimization-based and learning-based methods. Different priors are integrated into the ill-posed creation pipeline from sparse in-the-wild images. These include learning light transport via synthetic datasets, incorporating a procedural graph database for high-quality material optimization, and distilling knowledge from a generative model trained on large-scale real-world images and fine-tuned on a small, customized image set to generate specific high-quality textures.

As a result, we have developed frameworks capable of generating explicit, relightable 3D assets or scene configurations with editability for XR applications. Additionally, we propose a relighting network designed to enhance immersive XR experiences through portrait relighting techniques. For future work, we aim to improve the speed, intrinsic decomposition, and 3D consistency and coherency of the photorealistic 3D content creation pipeline, as detailed in the Section 7.1. We believe this dissertation lays a solid foundation for advancing practical photorealistic content creation pipelines from sparse in-the-wild image sets.

## 7.1 Future Work

In the optimization approach, a differentiable renderer is implemented in the pipeline. For the sake of rendering speed, it is typically assumed that the lighting is an infinitely distant

environment illumination, allowing for rendering with a differentiable rasterizer. However, this assumption fails in scenarios where the light sources are closer, such as lamps, which are usually not infinitely far away. In these cases, differentiable Monte Carlo path tracing and denoising approaches are necessary but require more resources and time to compute. A practical direction for improving speed is to explore learning-based solutions for content creation. The key challenge lies in finding a balance between speed and quality. By leveraging machine learning techniques, we can potentially achieve faster rendering times while maintaining high-quality results.

Inverse rendering or intrinsic image decomposition problems are ill-posed, especially when dealing with a sparse set of input images. Addressing lighting baked into materials or textures remains an ongoing research challenge. However, the emergence of diffusion models has been shown to significantly improve intrinsic decomposition by providing better guidance after fine-tuning a foundation model with synthetic data [107]. Conversely, for image synthesis, foundation models trained on large-scale datasets can accurately replicate physical properties, such as reflections on water or floors and transparency, resulting in photorealistic outputs. Thus, diffusion models could be utilized as simulation machines, offering training guidance for an inverse rendering pipeline. This approach can mitigate the quality loss caused by using less photorealistic synthetic data as supervision for inverse rendering tasks.

While 2D data contains billions of training examples (e.g., LAION-5B [192]), the well-known Janus problem or 3-face problem is inevitable when applying 2D diffusion models to 3D tasks. Currently, the largest 3D dataset, Objaverse-XL [42], contains only 10 million examples, which is much smaller in scale compared to 2D datasets. Recently, OpenAI released a text-to-video generation model that demonstrates unprecedented video generation quality. Video data is easier to acquire compared to 3D data, and the 3D consistency learned implicitly from video data can be beneficial for 3D tasks. Combining real-world image data with video data can enhance the generalization of models trained solely on 3D datasets. The potential next step is to leverage the strengths of video diffusion models to improve 3D generation by incorporating the 3D consistency learned implicitly from video data.

# Bibliography


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

[2] Blender. http://www.blender.org.

[3] Nvidia OptiX. https://developer.nvidia.com/optix.

[4] Substance Share. https://share.substance3d.com/.

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

[6] Badour AlBahar, Shunsuke Saito, Hung-Yu Tseng, Changil Kim, Johannes Kopf, and Jia-Bin Huang. Single-image 3d human digitization with shape-guided diffusion. In *SIGGRAPH Asia*, 2023.

[7] Xiang An, Xuhan Zhu, Yang Xiao, Lan Wu, Ming Zhang, Yuan Gao, Bin Qin, Debing Zhang, and Fu Ying. Partial fc: Training 10 million identities on a single machine. In *Arxiv 2010.05222*, 2020.

[8] Amir Atapour-Abarghouei and Toby P Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. Time-resolved 3D capture of non-stationary gas flows. *ACM ToG*, 27(5):132:1–132:9, December 2008.

[10] Armen Avetisyan, Tatiana Khanova, Christopher Choy, Denver Dash, Angela Dai, and Matthias Nießner. Scenecad: Predicting object alignments and layouts in rgb-d scans. *arXiv preprint arXiv:2003.12622*, 2020.

[11] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[12] Stephen T Barnard and William B Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4):333–340, 1980.

[13] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013.

[14] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[15] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2010.

[16] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] Ben-Ezra and Nayar. What does motion reveal about transparency? In *ICCV*, pages 1025–1032 vol.2, 2003.

[18] Paul J. Besl and Neil D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

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

[20] Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. Mesh2tex: Generating mesh textures from image queries. *arXiv preprint arXiv:2304.05868*, 2023.

[21] G. Cai, K. Yan, Z. Dong, I. Gkioulekas, and S. Zhao. Physics-based inverse rendering using combined implicit and explicit geometries. *Computer Graphics Forum*, 41(4):129–138, 2022.

[22] Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4169–4181, 2023.

[23] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022.

[24] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[25] Visesh Chari and Peter Sturm. A theory of refractive photo-light-path triangulation. In *CVPR*, pages 1438–1445, Washington, DC, USA, 2013. IEEE Computer Society.

[26] Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Inverse transport networks. *CoRR*, abs/1809.10820, 2018.

[27] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023.

[28] Guanying Chen, Kai Han, and Kwan-Yee K Wong. Learning transparent object matting. *IJCV*, 127(10):1527–1544, 2019.

[29] Kang Chen, Kun Xu, Yizhou Yu, Tian-Yi Wang, and Shi-Min Hu. Magic decorator: Automatic material suggestion for indoor digital scenes. *ACM Trans. Graph.*, 34(6), October 2015.

[30] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023.

[31] T. Chen, H. P. A. Lensch, C. Fuchs, and H. Seidel. Polarization and phase-shifting for 3D scanning of translucent objects. In *CVPR*, pages 1–8, June 2007.

[32] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *Advances in Neural Information Processing Systems*, 35:30923–30936, 2022.

[33] Zhiqin Chen, Kangxue Yin, and Sanja Fidler. Auv-net: Learning aligned uv maps for texture transfer and synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1465–1474, 2022.

[34] Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *arXiv preprint arXiv:2107.06278*, 2021.

[35] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 628–644. Springer, 2016.

[36] Yung-Yu Chuang, Douglas E. Zongker, Joel Hindorff, Brian Curless, David H. Salesin, and Richard Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *SIGGRAPH*, pages 121–130, 2000.

[37] Z. Cui, J. Gu, B. Shi, P. Tan, and J. Kautz. Polarimetric multi-view stereo. In *CVPR*, pages 369–378, July 2017.

[38] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[39] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7830–7839, 2020.

[40] Paul Debevec. The light stages and their applications to photoreal digital actors. *ACM SIGGRAPH Asia Technical Briefs*, 2012.

[41] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[42] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] 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 (ToG)*, 37(4):1–15, 2018.

[44] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Flexible SVBRDF capture with a multi-image deep network. *Computer Graphics Forum*, 38(4):1–13, 2019.

[45] Abdallah Dib, Gaurav Bharaj, Junghyun Ahn, Cédric Thébault, Philippe Gosselin, Marco Romeo, and Louis Chevallier. Practical face reconstruction via differentiable ray tracing. In *Computer Graphics Forum*, 2021.

[46] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[47] Qi Duan, Jianfei Cai, and Jianmin Zheng. Compressive environment matting. *Vis. Comput.*, 31(12):1587–1600, December 2015.

[48] Aysegul Dundar, Ming-Yu Liu, Ting-Chun Wang, John Zedlewski, and Jan Kautz. Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation. *arXiv preprint arXiv:1807.09384*, 2018.

[49] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[50] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. *arXiv preprint arXiv:2303.17015*, 2023.

[51] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Orcun Cetintas, Riccardo Gasparini, Aljoša Ošep, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. Motsynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[52] Clara Fernandez-Labrador, Alejandro Perez-Yus, Gonzalo Lopez-Nicolas, and Jose J Guerrero. Layouts from panoramic images with geometry and deep learning. *IEEE Robotics and Automation Letters*, 3(4):3153–3160, 2018.

[53] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129:3313–3337, 2021.

[54] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2022.

[55] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Trans. Graph.*, 38(4), July 2019.

[56] 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 (TOG)*, 38(4):134, 2019.

[57] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

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

[59] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[60] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. Arnold: A brute-force production path tracer. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2018.

[61] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9785–9795, 2019.

[62] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[63] Paul Graham, Borom Tunwattanapong, Jay Busch, Xueming Yu, Andrew Jones, Paul Debevec, and Abhijeet Ghosh. Measurement-based synthesis of facial microgeometry. In *Computer Graphics Forum*, 2013.

[64] James Gregson, Michael Krimerman, Matthias B. Hullin, and Wolfgang Heidrich. Stochastic tomography and its applications in 3D imaging of mixing fluids. *ACM ToG*, 31(4):52:1–52:10, July 2012.

[65] 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 Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2019.

[66] Yu Guo, Milos Hasan, Ling-Qi Yan, and Shuang Zhao. A bayesian inference framework for procedural material parameter estimation. *CoRR*, abs/1912.01067, 2019.

[67] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. Materialgan: reflectance capture using a generative svbrdf model. *arXiv preprint arXiv:2010.00114*, 2020.

[68] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023.

[69] Kai Han, Kwan-Yee K. Wong, and Miaomiao Liu. Dense reconstruction of transparent objects by altering incident light paths through refraction. *Int. J. Comput. Vision*, 126(5):460–475, May 2018.

[70] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[71] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv:2206.03380*, 2022.

[72] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. *Advances in Neural Information Processing Systems*, 35:22856–22869, 2022.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[74] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7498–7507, 2020.

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

[76] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[77] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[78] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[79] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[80] Andrew Hou, Michel Sarkis, Ning Bi, Yiying Tong, and Xiaoming Liu. Face relighting with geometrically consistent shadows. *arXiv preprint arXiv:2203.16681*, 2022.

[81] Andrew Hou, Ze Zhang, Michel Sarkis, Ning Bi, Yiying Tong, and Xiaoming Liu. Towards high fidelity face relighting with realistic shadows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[82] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[83] Yiwei Hu, Julie Dorsey, and Holly Rushmeier. A novel framework for inverse procedural texture modeling. *ACM Trans. Graph.*, 38(6), November 2019.

[84] Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. An inverse procedural modeling pipeline for svbrdf maps. *arXiv preprint arXiv:2109.06395*, 2021.

[85] Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. Holistic 3d scene parsing and reconstruction from a single rgb image. In *Proceedings of the European conference on computer vision (ECCV)*, pages 187–203, 2018.

[86] C. P. Huynh, A. Robles-Kelly, and E. Hancock. Shape and refractive index recovery from single-view polarisation images. In *CVPR*, pages 1229–1236, June 2010.

[87] Ivo Ihrke, Kiriakos Kutulakos, Hendrik Lensch, Marcus Magnor, and Wolfgang Heidrich. Transparent and specular object reconstruction. *Comput. Graph. Forum*, 29:2400–2426, 12 2010.

[88] Ivo Ihrke and Marcus Magnor. Image-based tomographic reconstruction of flames. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 365–373, Goslar Germany, Germany, 2004. Eurographics Association.

[89] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5134–5143, 2017.

[90] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Material memex: Automatic material suggestions for 3d objects. *ACM Transactions on Graphics (TOG)*, 31(6):1–8, 2012.

[91] 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*, 2001.

[92] Y. Ji, J. Ye, and J. Yu. Reconstructing gas flows using light-path approximation. In *CVPR*, pages 2507–2514, June 2013.

[93] Yoshihiro Kanamori and Yuki Endo. Relighting humans: Occlusion-aware inverse rendering for full-body human images. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2018.

[94] Yoshihiro Kanamori and Yuki Endo. Relighting humans: Occlusion-aware inverse rendering for full-body human images. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2018.

[95] 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.

[96] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4:3, 2013.

[97] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18423–18433, 2023.

[98] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[99] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)*, 30(6):1–12, 2011.

[100] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics (TOG)*, 33(3):1–15, 2014.

[101] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-free score distillation. *arXiv preprint arXiv:2310.17590*, 2023.

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

[103] Zhanghan Ke, Jiayu Sun, Kaican Li, Qiong Yan, and Rynson W.H. Lau. MODNet: Real-time trimap-free portrait matting via objective decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[104] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[105] J. Kim, I. Reshetouski, and A. Ghosh. Acquiring axially-symmetric transparent objects using single-view transmission imaging. In *CVPR*, pages 1484–1492, July 2017.

[106] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[107] Peter Kocsis, Vincent Sitzmann, and Matthias Nie{ssner. Intrinsic image diffusion for single-view material estimation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[108] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. In *ACM SIGGRAPH 2007 papers*, pages 2–es. 2007.

[109] K. N. Kutulakos and E. Steger. A theory of refractive and specular 3D shape by light-path triangulation. In *ICCV*, volume 2, pages 1448–1455 Vol. 2, Oct 2005.

[110] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 2000.

[111] Kiriakos N. Kutulakos and Eron Steger. A theory of refractive and specular 3D shape by light-path triangulation. *IJCV*, 76(1):13–29, 2008.

[112] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003.

[113] Manuel Lagunas, Xin Sun, Jimei Yang, Ruben Villegas, Jianming Zhang, Zhixin Shu, Belen Masia, and Diego Gutierrez. Single-image full-body human relighting. In *Eurographics Symposium on Rendering*, 2021.

[114] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

[115] Sangrok Lee, Eunsoo Park, Hongsuk Yi, and Sang Hun Lee. Strdan: Synthetic-to-real domain adaptation network for vehicle re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 2020.

[116] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, 2000.

[117] Chen Li and Gim Hee Lee. From synthetic to real: Unsupervised domain adaptation for animal pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[118] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[119] 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.

[120] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM ToG (SIGGRAPH Asia)*, 37(6):222:1 – 222:11, 2018.

[121] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2475–2484, 2020.

[122] Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. Materials for masses: Svbrdf acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–87, 2018.

[123] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[124] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM ToG (SIGGRAPH Asia)*, 37(6):269:1 – 269:11, 2018.

[125] Zhengqin Li, Yu-Ying Yeh, and Manmohan Chandraker. Through the looking glass: Neural 3d reconstruction of transparent shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1262–1271, 2020.

[126] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Miloš Hašan, Ravi Ramamoorthi, and Manmohan Chandraker. OpenRooms: An end-to-end open framework for photorealistic indoor scene datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[127] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[128] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2261–2269, 2017.

[129] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020.

[130] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2018.

[131] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM ToG (SIGGRAPH Asia)*, 38(4):65:1–65:14, 2019.

[132] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[133] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021.

[134] Yiwei Ma, Xiaoqing Zhang, Xiaoshuai Sun, Jiayi Ji, Haowei Wang, Guannan Jiang, Weilin Zhuang, and Rongrong Ji. X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2749–2760, 2023.

[135] BR Mallikarjun, Ayush Tewari, Abdallah Dib, Tim Weyrich, Bernd Bickel, Hans Peter Seidel, Hanspeter Pfister, Wojciech Matusik, Louis Chevallier, Mohamed A Elgharib, and Christian Theobalt. Photoapp: Photorealistic appearance editing of head portraits. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2021.

[136] Steve Marschner. Inverse rendering for computer graphics. 1998.

[137] Wojciech Matusik, Hanspeter Pfister, Remo Ziegler, Addy Ngan, and Leonard McMillan. Acquisition and rendering of transparent and refractive objects. In *Eurographics Workshop on Rendering*, EGRW '02, pages 267–278, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

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

[139] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6315–6324, 2018.

[140] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[141] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023.

[142] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021.

[143] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022.

[144] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[145] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[146] Gene S Miller and CR Hoffman. Illumination and reflection maps. In *ACM SIGGRAPH*, volume 4, 1984.

[147] D. Miyazaki and K. Ikeuchi. Inverse polarization raytracing: estimating surface shapes of transparent objects. In *CVPR*, volume 2, pages 910–917 vol. 2, June 2005.

[148] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022.

[149] Joep Moritz, Stuart James, Tom SF Haines, Tobias Ritschel, and Tim Weyrich. Texture stationarization: Turning photos into tileable textures. In *Computer graphics forum*, volume 36, pages 177–188. Wiley Online Library, 2017.

[150] N. J. W. Morris and K. N. Kutulakos. Dynamic refraction stereo. In *ICCV*, volume 2, pages 1573–1580 Vol. 2, Oct 2005.

[151] N. J. W. Morris and K. N. Kutulakos. Reconstructing the surface of inhomogeneous transparent scenes by scatter-trace photography. In *ICCV*, pages 1–8, Oct 2007.

[152] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

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

[154] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022.

[155] Koki Nagano, Graham Fyffe, Oleg Alexander, Jernej Barbič, Hao Li, Abhijeet Ghosh, and Paul Debevec. Skin microstructure deformation with displacement map convolution. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2015.

[156] Koki Nagano, Huiwen Luo, Zejian Wang, Jaewoo Seo, Jun Xing, Liwen Hu, Lingyu Wei, and Hao Li. Deep face normalization. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2019.

[157] Thomas Nestmeyer, Jean-FranÃ§ois Lalonde, Iain Matthews, and Andreas M Lehrmann. Learning physics-guided face relighting under directional light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[158] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 55–64, 2020.

[159] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and lighting reconstruction for complex indoor scenes with texture-space differentiable rendering. 2021.

[160] Rohit Pandey, Sergio Orts Escolano, Chloe Legendre, Christian Haene, Sofien Bouaziz, Christoph Rhemann, Paul Debevec, and Sean Fanello. Total relighting: learning to relight portraits for background replacement. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2021.

[161] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[162] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), November 2018.

[163] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[164] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[165] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. *Computer Graphics Forum*, 22(4):663–687, 2003.

[166] Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi. Learning generative models of textured 3d meshes from real-world images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13879–13889, 2021.

[167] Pieter Peers and Philip Dutré. Wavelet environment matting. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 157–166, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

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

[169] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *tog*, 2019.

[170] Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. Free-viewpoint indoor neural relighting from multi-view stereo. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2021.

[171] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[172] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[173] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023.

[174] Y. Qian, M. Gong, and Y. Yang. Frequency-based environment matting by compressive sensing. In *ICCV*, pages 3532–3540, Dec 2015.

[175] Y. Qian, M. Gong, and Y. Yang. Stereo-based 3d reconstruction of dynamic fluid surfaces by global optimization. In *CVPR*, pages 6650–6659, July 2017.

[176] Yiming Qian, Minglun Gong, and Yee-Hong Yang. 3d reconstruction of transparent objects with position-normal consistency. In *CVPR*, pages 4369–4377, 06 2016.

[177] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. volume 106, page 107404, 2020.

[178] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[179] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv:2303.13508*, 2023.

[180] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[181] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[182] Fitsum Reda, Robert Pottorff, Jon Barker, and Bryan Catanzaro. flownet2-pytorch: Pytorch implementation of flownet 2.0: Evolution of optical flow estimation with deep networks. https://github.com/NVIDIA/flownet2-pytorch, 2017.

[183] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery.

[184] Stephan R Richter, Hassan Abu Al Haija, and Vladlen Koltun. Enhancing photorealism enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[185] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[186] Carlos Rodriguez-Pardo and Elena Garces. Seamlessgan: Self-supervised synthesis of tileable texture maps. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[187] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[188] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[189] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[190] Shen Sang and Manmohan Chandraker. Single-shot neural relighting and svbrdf estimation. In *ECCV*, 2020.

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

[192] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[193] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, volume 1, pages 519–528, June 2006.

[194] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.2.1.

[195] Soumyadip Sengupta, Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M Seitz. A light stage on every desk. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[196] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8598–8607, 2019.

[197] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D Castillo, and David W Jacobs. Sfsnet: Learning shape, reflectance and illuminance of facesin the wild'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6296–6305, 2018.

[198] Qi Shan, Sameer Agarwal, and Brian Curless. Refractive height fields from single and multiple images. In *CVPR*, pages 286–293, 06 2012.

[199] Prafull Sharma, Varun Jampani, Yuanzhen Li, Xuhui Jia, Dmitry Lagun, Fredo Durand, William T Freeman, and Mark Matthews. Alchemist: Parametric control of material properties with diffusion models. *arXiv preprint arXiv:2312.02970*, 2023.

[200] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: Differentiable material graphs for procedural material capture. *ACM Trans. Graph.*, 39(6):1–15, December 2020.

[201] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model, 2023.

[202] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.

[203] YiChang Shih, Sylvain Paris, Connelly Barnes, William T. Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2014.

[204] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[205] Zhixin Shu, Sunil Hadap, Eli Shechtman, Kalyan Sunkavalli, Sylvain Paris, and Dimitris Samaras. Portrait lighting transfer using a mass transport approach. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2017.

[206] 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.

[207] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *European Conference on Computer Vision*, pages 72–88. Springer, 2022.

[208] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[209] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[210] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[211] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[212] Jonathan Stets, Zhengqin Li, Jeppe Revall Frisvad, and Manmohan Chandraker. Single-shot analysis of refractive shape using convolutional neural networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 995–1003. IEEE, 2019.

[213] Jonathan Dyssel Stets, Alessandro Dal Corso, Jannik Boll Nielsen, Rasmus Ahrenkiel Lyngby, Sebastian Hoppe Nesgaard Jensen, Jakob Wilm, Mads Brix Doest, Carsten Gundlach, Eythor Runar Eiriksson, Knut Conradsen, Anders Bjorholm Dahl, Jakob Andreas Bærentzen, Jeppe Revall Frisvad, and Henrik Aanæs. Scene reassembly after multi-modal digitization and pipeline evaluation using photorealistic rendering. *Appl. Optics*, 56(27):7679–7690, 2017.

[214] Cheng Sun, Guangyan Cai, Zhengqin Li, Kai Yan, Cheng Zhang, Carl Marshall, Jia-Bin Huang, Shuang Zhao, and Zhao Dong. Neural-pbir reconstruction of shape, material, and illumination. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[215] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1047–1056, 2019.

[216] Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul E Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2019.

[217] Tiancheng Sun, Kai-En Lin, Sai Bi, Zexiang Xu, and Ravi Ramamoorthi. Nelf: Neural light-transport field for portrait view synthesis and relighting. In *Eurographics Symposium on Rendering*, 2021.

[218] Daichi Tajima, Yoshihiro Kanamori, and Yuki Endo. Relighting humans in the wild: Monocular full-body human relighting with domain adaptation. In *Computer Graphics Forum*, 2021.

[219] K. Tanaka, Y. Mukaigawa, H. Kubo, Y. Matsushita, and Y. Yagi. Recovering transparent shape from time-of-flight distortion. In *CVPR*, pages 4387–4395, June 2016.

[220] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020.

[221] Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1274–1283, 2017.

[222] Anh Thai, Stefan Stojanov, Vijay Upadhya, and James M Rehg. 3d reconstruction of novel object shapes from single images. In *2021 International Conference on 3D Vision (3DV)*, pages 85–95. IEEE, 2021.

[223] Thomas V Thompson, Ernest J Petti, and Chuck Tappan. Xgen: arbitrary primitive generator. In *ACM SIGGRAPH 2003 Sketches & Applications*. 2003.

[224] Borislav Trifonov, Derek Bradley, and Wolfgang Heidrich. Tomographic reconstruction of transparent objects. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH, New York, NY, USA, 2006. ACM.

[225] C. Tsai, A. Veeraraghavan, and A. C. Sankaranarayanan. What does a single light-ray reveal about a transparent object? In *ICIP*, pages 606–610, Sep. 2015.

[226] Giuseppe Vecchio, Simone Palazzo, and Concetto Spampinato. Surfacenet: Adversarial svbrdf estimation from a single image. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[227] Madhawa Vidanapathirana, Qirui Wu, Yasutaka Furukawa, Angel X. Chang, and Manolis Savva. Plan2scene: Converting floorplans to 3d scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10733–10742, June 2021.

[228] Dan Wang, Xinrui Cui, Xun Chen, Zhengxia Zou, Tianyang Shi, Septimiu Salcudean, Z Jane Wang, and Rabab Ward. Multi-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5722–5731, 2021.

[229] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023.

[230] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34, 2021.

[231] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[232] Tuanfeng Y. Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy J. Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph.*, 35(6), November 2016.

[233] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[234] Zhibo Wang, Xin Yu, Ming Lu, Quan Wang, Chen Qian, and Feng Xu. Single image portrait relighting via explicit multiple reflectance channel modeling. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2020.

[235] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 2004.

[236] 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 (Proc. SIGGRAPH)*, 2005.

[237] G. Wetzstein, D. Roodnick, W. Heidrich, and R. Raskar. Refractive shape from light field distortion. In *ICCV*, pages 1180–1186, Nov 2011.

[238] Yonatan Wexler, Andrew Fitzgibbon, and Andrew Zisserman. Image-based environment matting. In *CVPR*, pages 279–290, 01 2002.

[239] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, and Markus Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2006.

[240] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[241] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Matthew Johnson, Jingjing Shen, Nikola Milosavljević, Daniel Wilde, Stephan Garbin, Toby Sharp, Ivan Stojiljković, Thomas J. Cashman, and Julien Valentin. 3d face reconstruction with dense landmarks. *arXiv preprint arXiv:2204.02776*, 2022.

[242] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical engineering*, 19(1):139–144, 1980.

[243] Bojian Wu, Yang Zhou, Yiming Qian, Minglun Cong, and Hui Huang. Full 3d reconstruction of transparent objects. *ACM ToG*, 37(4):103:1–103:11, July 2018.

[244] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 30, 2017.

[245] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016.

[246] Zhaohui Wu, Zhong Zhou, Delei Tian, and Wei Wu. Reconstruction of three-dimensional flame with color temperature. *Vis. Comput.*, 31(5):613–625, May 2015.

[247] Haozhe Xie, Hongxun Yao, Shengping Zhang, Shangchen Zhou, and Wenxiu Sun. Pix2vox++: multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision*, 128(12):2919–2935, 2020.

[248] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems*, 32, 2019.

[249] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*, 2022.

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

[251] K. Yan, F. Luan, M. Hašan, T. Groueix, V. Deschaintre, and S. Zhao. Psdr-room: Single photo to scene using differentiable rendering. In *ACM SIGGRAPH Asia 2023 Conference Proceedings*, 2023.

[252] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019.

[253] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent MVSNet for high-resolution multi-view stereo depth inference. In *CVPR*, June 2019.

[254] Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, and Manmohan Chandraker. Photoscene: Photorealistic material and lighting transfer for indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[255] Yu-Ying Yeh, Koki Nagano, Sameh Khamis, Jan Kautz, Ming-Yu Liu, and Ting-Chun Wang. Learning to relight portrait images via a virtual light stage and synthetic-to-real adaptation. *ACM Transactions on Graphics (TOG)*, 2022.

[256] S. Yeung, T. Wu, C. Tang, T. F. Chan, and S. Osher. Adequate reconstruction of transparent objects on a shoestring budget. In *CVPR*, pages 2513–2520, June 2011.

[257] S. Yeung, T. Wu, C. Tang, T. F. Chan, and S. J. Osher. Normal estimation of a transparent object using a video. *PAMI*, 37(4):890–897, April 2015.

[258] Sai-Kit Yeung, Chi-Keung Tang, Michael S. Brown, and Sing Bing Kang. Matting and compositing of transparent and refractive objects. *ACM ToG (SIGGRAPH)*, 30(1):2:1–2:13, 2011.

[259] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[260] Xin Yu, Peng Dai, Wenbo Li, Lan Ma, Zhengzhe Liu, and Xiaojuan Qi. Texture generation on 3d meshes with point-uv diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4206–4216, 2023.

[261] Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3d scene understanding from a single image with implicit representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8833–8842, 2021.

[262] 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.

[263] Longwen Zhang, Qixuan Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Neural video portrait relighting in real-time via consistency modeling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[264] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[265] Mingjie Zhang, Xing Lin, Mohit Gupta, Jinli Suo, and Qionghai Dai. Recovering scene geometry under wavy fluid via distortion and defocus analysis. In *ECCV*, volume 8693, pages 234–250, 09 2014.

[266] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[267] 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 (Proc. SIGGRAPH)*, 2021.

[268] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[269] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. *Advances in neural information processing systems*, 31, 2018.

[270] Xuaner Zhang, Jonathan T Barron, Yun-Ta Tsai, Rohit Pandey, Xiuming Zhang, Ren Ng, and David E Jacobs. Portrait shadow manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2020.

[271] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[272] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W Jacobs. Deep single-image portrait relighting. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

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

[274] Xilong Zhou, Milos Hasan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari. Tilegen: Tileable, controllable material generation and capture. In *SIGGRAPH Asia 2022 conference papers*, pages 1–9, 2022.

[275] Jiayuan Zhu and Yee-Hong Yang. Frequency-based environment matting. In *Pacific Graphics*, pages 402–410, 2004.

[276] Jingsen Zhu, Fujun Luan, Yuchi Huo, Zihao Lin, Zhihua Zhong, Dianbing Xi, Rui Wang, Hujun Bao, Jiaxiang Zheng, and Rui Tang. Learning-based inverse rendering of complex indoor scenes with differentiable monte carlo raytracing. In *SIGGRAPH Asia 2022 Conference Papers*. ACM, 2022.

[277] Douglas E. Zongker, Dawn M. Werner, Brian Curless, and David H. Salesin. Environment matting and compositing. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 205–214, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[278] X. Zuo, C. Du, S. Wang, J. Zheng, and R. Yang. Interactive visual hull refinement for specular and transparent object surface reconstruction. In *ICCV*, pages 2237–2245, Dec 2015.