A dissertation

submitted in partial fulfillment of the

requirements for the degree of

University of Washington

Reading Committee:

Program Authorized to Offer Degree:

University of Washington

Abstract

Scene Rerendering

Keunhong Park

Chair of the Supervisory Committee:
Robert E. Dinning Professor Steven M. Seitz
Computer Science and Engineering

Taking a good photograph can be a time-consuming process, and it usually takes several attempts to capture a moment correctly. This difficulty stems from the many factors that make up a photo, such as framing, perspective, exposure, focus, or subject pose. Getting even one of these factors wrong can spoil a picture, even if the rest are perfect. To make matters worse, many of these factors are often out of our control; for example, a wind gust may displace the subject's hair, or a bird may fly by and occlude the shot. What if we could go back and fix some of these aspects? In my thesis, I explore techniques for "*scene rerendering*" which enable rich modification of media after capture.

First, I propose *Nerfies*, the first method capable of photo-realistically reconstructing a non-rigidly deforming scene using photos and videos captured casually from mobiles phones. Nerfies augments neural radiance fields (NeRF) by optimizing an additional continuous volumetric deformation field that warps each observed point into a canonical 5D NeRF. I show that these NeRF-like deformation fields are prone to local minima and propose a coarse-to-fine optimization method that allows for more robust optimization. By adopting principles from geometry processing and physical simulation to NeRF-like models, I also propose an elastic regularization of the deformation field that further improves robustness. I demonstrate how Nerfies can turn casually captured selfie photos/videos into deformable NeRF models that allow for photorealistic renderings of the subject from arbitrary viewpoints.

Deformation-based approaches such as Nerfies struggle to model changes in topology (e.g., slicing a lemon), as topological changes require a discontinuity in the deformation field, but these deformation fields are necessarily continuous. I address this limitation in *HyperNeRF*, in which I propose lifting NeRFs into a higher-dimensional space and by representing the 5D radiance field corresponding to each input image as a slice through this "hyperspace." This approach is inspired by level set methods, which model the evolution of surfaces as slices through a higher dimensional surface.

Next, I present *PhotoShape*, an approach that creates photorealistic, relightable 3D models automatically. PhotoShape automatically assigns high-quality, realistic appearance models to large-scale 3D shape collections. By generating many synthetic renderings, I train a convolutional neural network to classify materials in real photos and employ 3D-2D alignment techniques to transfer materials to different parts of each shape model. The key idea is to jointly leverage three types of online data – shape collections, material collections, and photo collections, using the photos as reference to guide the assignment of materials to shapes.

Finally, I show how we can exploit methods for scene rerendering to solve *inverse* problems. I propose *LatentFusion*, a framework for performing 3D reconstruction and rendering using a neural network. This neural network takes posed images of an object as input and can render it from any novel viewpoint. I show how LatentFusion can be used for 6D object pose estimation by optimizing the input pose as a free parameter using gradient descent. Also, since this method incorporates objects at inference time, it can perform pose estimation on unseen objects without additional training— an immense benefit over existing methods which require training a separate network for every new object.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

(a) captured photos (b) fixing framing (c) fixing exposure

Figure 1.1: Photographers often use software to edit their photographs after capture. For example, a photographer may (b) crop a photo or (c) change the exposure of a photo.

As we scurry through the bustling streets of New York or as our friends jump into a lake, we often have our cameras out, taking photos of the fleeting moment. We juggle countless camera settings as we take a picture in the heat of the moment. It is common to miss opportunities to capture a moment or end up with a bad photo due to incorrect settings.

When taking a photo, we must consider many different elements, such as framing, perspective, or subject pose. A central challenge of photography is quickly finding the right combination of these variables. Many modern tools seek to ease this challenge by enabling users to edit elements of a photograph *after* capture. For instance, you can change the crop of a photo (Figure 1.1b) or adjust the exposure (Figure 1.1c). Just as modern word processors have fundamentally changed how we write documents, the ability the edit images has greatly influenced how many people approach photography.

Image editors are increasingly powerful, but they are limited because they operate on images that are mere 2D arrays of colors and cannot edit elements that are tied to the underlying 3D geometry of the scene. For example, perspective is critical in portrait pho-

(a) Telephoto ($f = 58$mm, $d = 2.0$m) (b) Standard ($f = 26$mm, $d = 0.4$m) (c) Ultrawide ($f = 13$mm, $d = 0.2$m)

Figure 1.2: Three photos taken with different focal lengths (f) from different distances (d). Perspective is a crucial element in portrait photography that can determine whether a subject look flattering or goofy. Perspective cannot be easily changed after capture as it is determined by the actual position of the camera relative to the subject.

tography. Figure 1.2 shows three different photos taken from different distances but with the lens zoomed such that the apparent size of the subject remains constant. Note how the photo from the closest distance (Figure 1.2c) shows overly exaggerated facial features such as a large nose. What if you want to edit the perspective of a picture? This would require changing the physical, 3D position of the camera at capture.

In this thesis, I explore the methods for *scene rerendering* in order to lift images from their 2D confines to enable more fundamental changes to their structure and appearance. In particular, my contributions include: ① a photo-realistic 3D reconstruction method that turns short videos or photo collections into a 3D scene that can be rendered from any camera perspective; ② a method to automatically create *relightable* 3D textured models, which can produce photo-realistic renderings with arbitrary lighting; and ③ a differentiable rendering pipeline that exploits scene rerendering to solve inverse problems such as pose estimation, supporting applications in robotics.

(a) selfie capture (b) input photos (c) novel views with longer focal length (d) novel view depth

Figure 1.2: We reconstruct photo-realistic *nerfies* from a user casually waving a mobile phone (a). Our system uses selfie photos/videos (b) to produce a free-viewpoint representation with accurate renders (c) and geometry (d). Nerfies can produce portraits free of perspective distortions usually present in selfie portraits by simply rendering the scene from a distant camera.

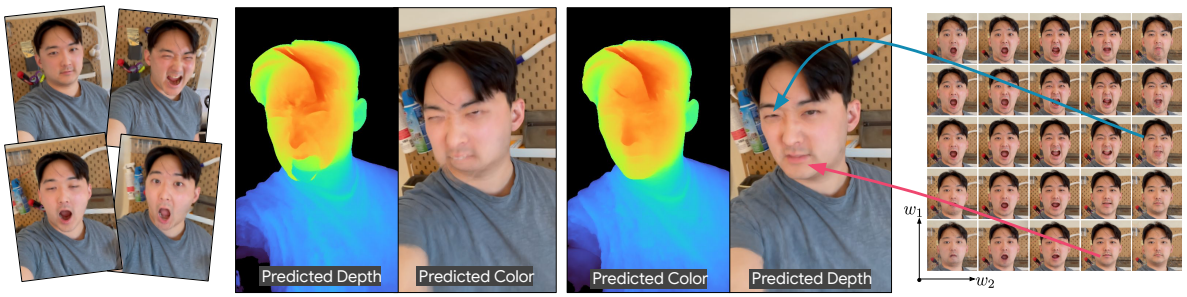
1.1 Casual 3D Capture

Changing elements like perspective or subject pose requires a 3D representation of the underlying scene which we can *rerender* as needed. *3D reconstruction* is the problem of acquiring such a representation. While 3D reconstruction is a well-established topic of study, its technologies have been mostly out of reach for casual users. Current methods often require the use of complicated and expensive capture setups such as multiple synchronized cameras and lights [43, 53, 72, 127].

The desire for simpler capturing regimes has spurred interest in *casual* 3D capture methods [74–76, 107, 132, 142, 245]. Relevant methods attempt to relax assumptions, enable simpler workflows (e.g., waving a phone around), and eliminate the need for complicated and expensive hardware. Many works have made significant progress in casual 3D capture [75, 76, 142], but high-quality reconstruction of complex geometry and lighting appearances have remained elusive until the recent advent of Neural Radiance Fields (NeRF) [143].

1.1.1 Dynamic Scenes

In Chapter 3, I introduce Nerfies [159], a system for capturing photo-realistic 3D representations from casually captured videos or image collections. Nerfies extends NeRF to address



(a) Input Images (b) Nerfies $(\mathbf{c}, \sigma) = F(x, y, z)$ (c) HyperNeRF $(\mathbf{c}, \sigma) = F(x, y, z, w_1, w_2)$ (d) Hyper-Space

Figure 1.3: Nerfies [159] can capture non-static subjects, but often struggle in the presence of topological variation, as evidenced in (b). By modeling a family of shapes in a high dimensional space shown in (d), our HyperNeRF model is able to handle topological variation and thereby produce more realistic renderings as can be seen in (c).

a critical limitation: the assumption of a static scene. Our world is rarely static, and encoding movement within a scene is crucial especially for dynamic subjects such as people. In addition, we wish to be able to alter the *subject's pose*. Therefore, Nerfies provides a mechanism to encode movement within the scene and allows interpolation between different subject poses. Nerfies can render a scene from any camera and can therefore control elements such as *perspective*. Portraits taken using a selfie camera are often unflattering due to the perspective distortion of photos taken from up close (Figure 1.3b). Nerfies can produce visually pleasing portraits as if the photo was taken from afar (Figure 1.3c).

1.1.2 Topologically Varying Scenes

Many real-world motions involve changes in topology. Examples include cutting a lemon, or tearing a piece of paper. While not strictly a genus change, motions like closing your mouth cause changes in surface connectivity that can also be considered "topological" (e.g., Figure 1.3a). Such changes in topology often cause problems for algorithms that seek to reconstruct moving three dimensional scenes, as they cause motion discontinuities or singularities. Nerfies is an example of a method that cannot model such motion due to its use of a continuous deformation field.

In Chapter 4, I propose HyperNeRF [160], an extension of Nerfies to handle topologically

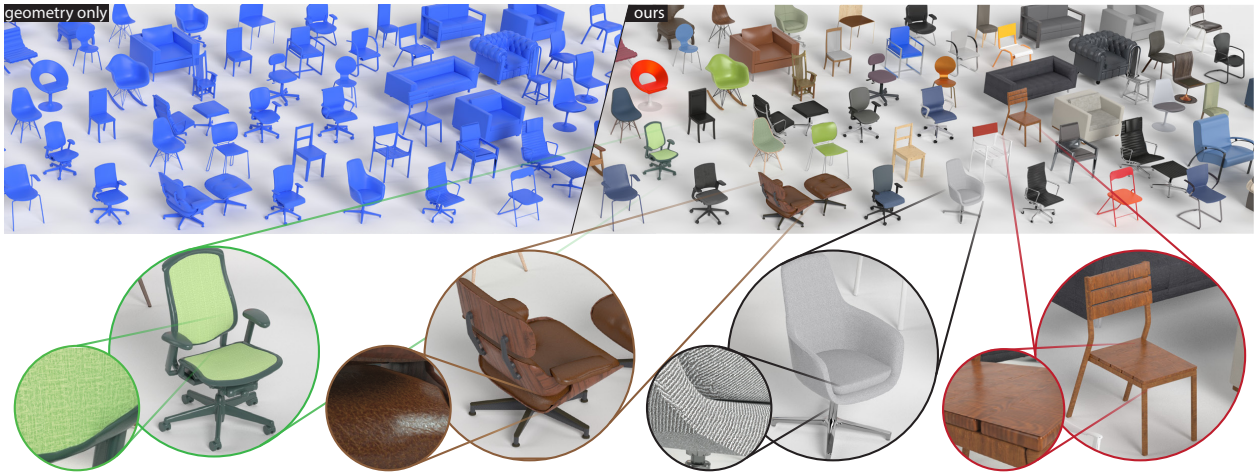


Figure 1.4: PhotoShape is a fully automated pipeline which creates *relightable* 3D models.

varying scenes. HyperNeRF takes inspiration from the classic approach of *level-set* methods. Pioneered in the late 1980s [153], level set methods model moving scenes as static objects in a higher dimensional ambient space (illustrated in Figure 1.3d), and topological changes as *smooth* (rather than discontinuous) transformations. HyperNeRF adapts the level set framework for Nerfies to generate photo-realistic, free-viewpoint renderings of objects undergoing changes in topology.

1.2 Relightable Models

§1.1 addresses scene-level capture, but for many applications such as augmented and virtual reality, we desire object-level 3D content which can blend into any virtual environment. In order for a 3D model to be realistically inserted into a virtual environment, it must be *relightable*, meaning each part of the 3D model must be associated with a material model which models how it reflects light. While there are many online repositories containing vast numbers of 3D models, the majority of these them do not include such material information.

In Chapter 5, I introduce PhotoShape [157], a fully-automatic pipeline capable for creating thousands of photo-realistic, relightable 3D models. PhotoShape matches images to

(a) Initial Depth (b) Initial RGB (c) Optimized Depth (d) Optimized RGB (e) Target RGB

Figure 1.6: The HyperNeRF model is invertible in the sense that input parameters such as the conditioning latent code can be optimized given an objective function. In this example, we initialize the model to an intial pose (b) and optimize the latent code to photometrically match the target RGB image (e). The resulting optimized rendering is shown in (d).

3D models of the same object, and then uses a neural network to classify image parts to materials models, represented as spatially varying BRDFs (SV-BRDFs). Since the system has correspondence between 3D object parts and image segments, it can then assign the materials to the 3D model, resulting in a model which can be re-lit using arbitrary lighting. Figure 1.4 shows examples of 3D models generated using this method.

1.3 Inverse Problems

Nerfies and HyperNeRF, introduced in §1.1, are examples of differentiable rendering pipelines. Namely, the entire image formation process from the 3D scene representation to the final rendered image is fully differentiable. Differentiable processes are useful as they can be used to solve challenging inverse problems such as state estimation. Differentiable processes can be inverted by performing gradient descent on any of the inputs with respect to some objective function. For example, Figure 1.6 shows an example where we invert the HyperNeRF pipeline by optimizing the conditioning latent code to fit the target RGB image best.

Chapter 6 describes LatentFusion [158], a framework specifically designed to exploit this

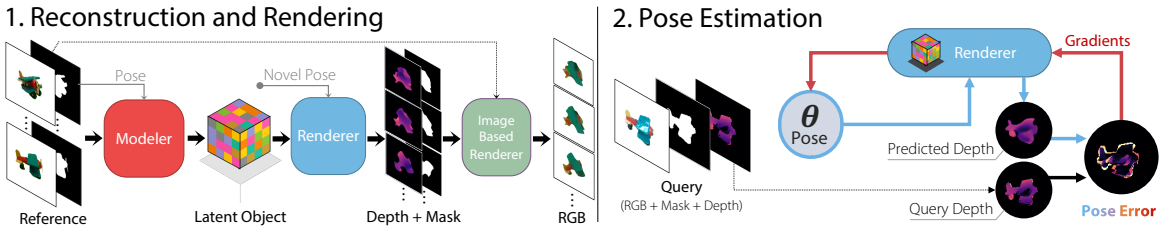


Figure 1.7: LatentFusion is an end-to-end differentiable reconstruction and rendering pipeline. This pipeline can be used to perform pose estimation on unseen objects using simple gradient updates in a render-and-compare fashion.

property of differentiable pipelines. LatentFusion performs both 3D reconstruction and rendering using a differentiable neural network (shown in Figure 1.7). This neural network takes posed images of an object as input, and can render it from any novel viewpoint. LatentFusion only needs to be trained once on synthetic data, and results in a category-agnostic model that can incorporate new, unseen objects at inference time.

A key application of LatentFusion is 6D object pose estimation. Since we can compute gradients through the rendering process, we can optimize the input pose as a free parameter using gradient descent. Also, since LatentFusion incorporates objects at inference time, it can perform pose estimation on unseen objects without additional training—a large benefit over existing methods which require training a separate network for every new object.

Chapter 2

RELATED WORK

In order to rerender a scene, there must be a representation of the scene that can be rendered from novel viewpoints. The most obvious and tedious method for creating such representations is by manually authoring them with 3D modeling software such as Blender or CAD. This is the method often used when creating assets for movies or video games. Manual authoring is a labor intensive process that must be performed by a highly skilled artist. In addition, it is incredibly difficult to accurately model all of the subtleties of real world scenes due to complex geometry and lighting effects. *3D reconstruction* methods capture objects and scenes using automated techniques. Not only are these methods much more accurate than hand-created models, but they are also much more labor efficient as the algorithms can do most of the heavy lifting and humans only has to follow simple capture protocols.

In this chapter, I conduct a survey of methods necessary for creating 3D representations that are the basis for rerenderable media. I introduce methods for 1) geometry reconstruction, different types of scene representations, and 3) methods for casual 3D capture.

2.1 Geometry Reconstruction

Authoring 3D content is a very labor intensive and difficult task. This fact has lead to the popularity of 3D reconstruction as a research topic. The goal is to simplify the process of modeling the *geometry* of an object or scene in the real world into something we can render on a computer, such as a 3D mesh.

Photogrammetry, in the context of computer vision, is the problem of creating a 3D representation of the real world through a collection of observations such as 2D images or other signals. In simpler terms, photogrammetry can be summarized as the task of turning

2D images into a 3D model. The problem can be split into two core sub-problems: (a) finding the location and parameters of the camera which took each image (localization), and (b) finding the 3D geometry which best explains the image (mapping).

2.1.1 Sparse Reconstruction and Camera Estimation

While there is no well established consensus on an exact taxonomy, offline methods which assume all input images are available simultaneously are called Structure-from-Motion (SfM), while online methods that update camera position and scene geometry as new image become available are usually called Simultaneous Localization and Mapping (SLAM).

Specific methods for SfM and SLAM are beyond the scope of this chapter, but there exist several robust open-source packages which implement robust systems based on the state of the art, such as Theia [201], Bundler [193], and COLMAP [178]. These methods provide the foundation for many 3D reconstruction techniques, and are used extensively in many of the works that are described below.

2.1.2 Dense Reconstruction

Given known camera positions and parameters, multi-view stereo (MVS) algorithms create a 3D representation of the scene [59, 67, 94]. While SfM and SLAM methods must recover both the camera parameters and the structure of the scene, MVS methods focus on the problem of recovering the accurate geometry of the scene. Since the objective of MVS is to recover the geometry of the scene, the resulting geometry is usually denser and more useful than those recovered by SfM and SLAM methods. For example, methods such as Jancosek and Padla *et al.* [94] and Labatut *et al.* [111] use Delaunay tetrahedralization to create 3D mesh representations of scenes from image collections. While MVS methods can result in impressive 3D reconstructions, they usually suffer from artifacts such as staircasing ([83, 83]) and lack of detail due to the use of discrete representations such as voxel grids.

2.1.3 Non-Rigid Reconstruction

Non-rigid reconstruction is the problem of reconstructing the geometry of a moving scene, e.g. of a person. A common approach in non-rigid reconstruction techniques is to decompose a scene into a canonical model of scene geometry (which is fixed across frames) and a deformation model that warps the canonical scene geometry to reproduce each input image. The difficulty of this task depends heavily on the inherent ambiguity of the problem formulation. Using only a monocular video stream is convenient and inexpensive, but also introduces significant ambiguity which must be ameliorated through the use of factorization [24] or regularization [207]. On the opposite end of the spectrum, complicated and expensive capture setups using multiple cameras and depth sensors can be used to overconstrain the problem, thereby allowing 3D scans to be registered and fused to produce high quality results [43, 53]. Machine learning techniques have been used effectively for non-rigid reconstruction, when applied to direct depth sensors [21, 177].

Several works have used learning techniques to solve for deformation based models of shape geometry [96, 152], though because these works model only geometry and not radiance, they cannot be applied to RGB image inputs and do not directly enable view synthesis. Yoon *et al*. [237] use a combination of multi-view cues as well as learned semantic priors in the form of monocular depth estimation to recover dynamic scenes from moving camera trajectories. In contrast, most NeRF-like models require no training data other than the input sequence being used as input. Neural Volumes [127] represents deformable scenes using a volumetric 3D voxel grid and a warp field, which are directly predicted by a convolutional neural network. As we will demonstrate, Neural Volumes's high fidelity output relies on the use of dozens of synchronized cameras, and does not generalize well to monocular image sequences. The Deformable NeRF technique we will introduce in Chapter 3 uses NeRF-like learned distortion fields alongside the radiance fields of [143] to recover "nerfies" of human subjects, and is capable of generating photorealistic synthesized views of a wide range of non-stationary subjects.

2.1.4 3D Scene Representations

Points and Surfaces Point-based and surface-based geometric representations are the dominant representation in computer vision and graphics. When we think of 3D geometry we usually think of it being encoded in something like a mesh, point cloud. While these representations are straightforward to reason about and are easy to render, they are unfortunately not very well suited when used in conjunction with optimization algorithms and struggle to handle complex phenomena such as thin structures, particle effects, and translucency.

Volumes Volumetric methods are popular when estimating the geometry of a scene, as they can be more easily updated incrementally by checking for photometric consistency. Volumetric methods such as voxel coloring [166, 181] and space carving [110] store occupancy and color information in a discrete voxel grid. While effective, these methods do not address the issue caused by thin structures or semi-transparent materials. Rather than just store occupancy information, more recent methods therefore store "softer" information such as probabilities [162, 176, 211, 212] or density [97, 127, 143].

Implicit Surfaces Another family of methods encode surfaces, but implicitly along the level-set of a function. For example, signed distance functions (SDF) store at each point in space the distance to the closest surface, where the sign determines whether the point is inside or outside the surface. Implicit surfaces are nice as they exhibit the nice optimization properties of volumetric methods, but encode a surface. They have therefore been popular in methods that merge range images into a canonical geometry, such as Curless *et al.* [44] and KinectFusion [148].

Density fields and implicit surfaces are more difficult to use than meshes, but it is possible to process them into more traditional representations using methods such as marching cubes [129].

Neural 3D Representations Volume rendering techniques, when combined with deep learning, have shown much promise in solving many of the problems encountered by traditional MVS methods. Tulsiani *et al*. [211] uses ray consistency to train a neural network to predict occupancy probabilities. Neural Volumes [127] uses a 3D convolutional neural network to merge multiple video streams into a single, coherent 3D representation that can be rendered from novel views. Neural Volumes can capture difficult cases such as fur or smoke that have long eluded traditional MVS methods. Recently the field has progressed rapidly with the introduction of methods that encode the geometry within the weights of a multi-layer perceptron. These methods, called coordinate-based models, take as input a spatial coordinate and output a single value. DeepSDF [156] stores truncated signed distance functions (tSDF) while Neural Radiance Fields (NeRF) [143] store volume density. Coordinate based models such as these are *continuous* and therefore avoid many issues that arise from discretization. These methods are incredibly robust during optimization and can handle complex phenomena such as semi-transparent materials and thin structures that have long eluded other approaches.

2.2 Image-based Rendering

An ideal 3D reconstruction method would be able to accurately model the complete geometry of a scene in addition to any material properties (albedo, specularity, etc.), and lighting. Creating an analytic model for all materials in the world and capturing subtle lighting effects is incredibly difficult. An increasingly large body of research therefore avoids explicitly modeling the 3D geometry or appearance of a scene, and instead uses clever techniques to directly compose input images into novel renderings. Such methods are often called image-based rendering (IBR) techniques.

Environment Maps Environment map based methods such as Quicktime VR [36] can be seen as an early precursor to modern image-based rendering techniques. Quicktime VR allowed users to move around a space by rotating projections of cylindrical panoramas and

formed a virtual environment by composing multiple panoramas. These panoramas were created by stitching together individual images taken with a camera mounted on a rotating tripod mount. These panoramas were expensive to create at the time, using many compute cycles on a Cray supercomputer. The biggest reason for this is likely the limited power of computers at the time, but it can also be attributed to the use of correlation-based image registration techniques that are more expensive than more modern techniques that use cheap features such as SIFT [130] or SURF [10]. Nowadays creating panoramas is incredibly easy — anyone can make a panorama from their mobile phone and it takes less than a second. Panoramas are a great example of how simply increasing the availability of an already existing technology have a significant impact on our everyday lives.

Nearest-Neighbors and Blending Quicktime VR also supported rendering objects in 3D by first capturing them on a turntable and then fetching the closest image to the desired target view. Showing the closest image means any intermediate viewpoints not captured could not be rendered, resulting a rough, jagged viewing experience. Works such as Chen *et al.* [37] and Werner *et al.* [224] therefore used estimated optical flow to interpolate input images into intermediate viewpoints. These methods only support rotation along a single axis since other changes in viewpoint would result in parallax which cannot be handled without depth information.

Light fields A light field describes the amount of light flowing in very direction in space, at every point in space. A light field is defined by the 5D plenoptic function [1] $P : (x, y, z, \theta, \phi) \rightarrow L$ which maps every point (x, y, z) and direction (θ, ϕ) to a radiance values (L). In a sense, the light field is a complete generalization of all 3D scene representations that can theoretically explain all optical phenomena. It is obviously impossible to know the amount of light flowing in all directions in all points in space so algorithms must make assumptions to turn light field rendering into a tractable problem.

Lumigraphs Gortler *et al*. [70] and Levoy *et al*. [112] both observed that the radiance of a scene does not change along a ray unless the ray is blocked. This means the dimensionality of the light field can be reduced from 5D to 4D, making the problem much more tractable. However, these methods still require a very dense sampling of input viewpoints captured using complicated capture setups, and only support images that are captured from cameras whose centers lie on a regularly sampled image plane.

View-dependent Texture Mapping (VDTM) Rather than assume a dense, regular sampling of input images, Debevec *et al*. [47] assumes that there is a good geometric model of the scene. The input images can then be projected onto the 3D model and rendered as texture maps. Since the geometry is known, this requires much less input images compared to lumigraph style methods.

Unstructured Lumigraphs [26] combines the lumigraph and VDTM approaches, using a globally consistent geometric *proxy* but also blends input images together to render novel viewpoints.

Learning-based Methods When rendering unseen views, it is inevitable that some information is missing from the input images that cannot be recovered with simple blending or interpolation. Deep learning is en effective tool for many tasks in computer vision, and novel-view synthesis is no exception. DeepStereo [57] predicted volumetric representations on a large dataset of posed images of streets. Kalantari *et al*. [98] used light field photos from a Lytro light field camera [64] to train a neural network to predict images from novel viewpoints. Stereo Magnification [245] introduced the multi-plane image (MPI) representation and learns only using stereo image pairs. The commonality between these methods is that they all train a neural network that provides a strong prior that can fill in the gaps in the presence of missing data. Some other methods, such as Deep Blending [76] learn how to blend between the input images with per-view geometry and a geometric proxy representation using deep learning as well. These methods can also be considered neural rendering

methods described in §2.3.

2.3 Neural Rendering

The nascent field of "neural rendering" aims, broadly, to use neural networks to render images of things. This is an emerging area of study that is changing rapidly, but progress in the field up through 2020 is well-documented in the survey report of [204].

2.3.1 Image-to-Image Translation

The dominant paradigm in this field has, until recently, been framing the task of synthesizing an image as a sort of "image to image translation" task, in which a neural network is trained to map some representation of a scene into an image of that scene [89]. This idea has been extended to incorporate tools and principles from the graphics literature, by incorporating reflectance or illumination models into the "translation" process [139, 198], or by using proxy geometries [5, 206] or conventional renderings [58, 102, 135, 141] as a harness with which neural rendering can then be guided. Though these techniques are capable of producing impressive results, they are often hampered by a lack of consistency across output renderings — when rendering is performed independently by a "black box" neural network, there is no guarantee that all renderings of scene will correspond to a single geometrically-consistent 3D world.

2.3.2 Neural Scene Representations

Research within neural rendering has recently begun to shift away from this "image to image translation" paradigm and towards a "neural scene representation" paradigm. Instead of "rendering" images using a black box neural network that directly predicts pixel intensities, scene-representation approaches use the weights of a neural network to directly model some aspect of the physical scene itself, such occupancy [140], distance [156], or a latent representation of appearance [189, 191].

Neural Radiance Fields The most effective approach within this paradigm has been constructing a neural radiance field (NeRF) of a scene, and using a conventional multilayer perceptron (MLP) to parameterize volumetric density and color as a function of spatial scene coordinates [143]. NeRF's usage of classical volumetric rendering techniques has many benefits in addition to enabling photorealistic view synthesis: renderings from NeRF must correspond to a single coherent model of geometry, and the gradients of volumetric rendering are well-suited to gradient-based optimization. Note that, in NeRF, a neural network is not used to render an image — instead, an analytical physics-based volumetric rendering engine is used to render a scene whose geometry and radiance happen to be parameterized by a neural network.

2.3.3 Neural Radiance Fields for Dynamic Scenes

Though NeRF produces compelling results on scenes in which all content is static, it fails catastrophically in the presence of moving objects. As such, a great deal of recent work has attempted to extend NeRF to support dynamic scenes. These NeRF variants can be divided into two distinct categories: *deformation-based* approaches apply a spatially-varying deformation to some canonical radiance field [159, 167, 209], and *modulation-based* approaches directly condition the radiance field of the scene on some property of the input image and modify it accordingly [114, 119, 229].

Deformation-Based NeRFs Deformation-based NeRF variants follow the tradition established by the significant body of research on non-rigid reconstruction [149], and map observations of the subject onto a template of that subject. Nerfies [**?**], D-NeRF [168], and NR-NeRF [209] all define a continuous deformation field which maps observation coordinates to canonical coordinates which are used to query a template NeRF. Because multiple observations of the subject are used to reconstruct a single canonical template, and only the deformation field is able to vary across images, this approach yields a well-constrained optimization problem similar to basic NeRF. Similar to how NeRF parameterized radiance

and density using a coordinate-based MLP, these deformation-based NeRF variants use an MLP to parameterize the deformation field of the scene, and are thereby able to recover detailed and complicated deformation fields. However, this use of a *continuous* deformation field means that these techniques are unable to model any topological variations (e.g., mouth openings) or transient effects (e.g, fire). Topological openings or closings require a discontinuity in the deformation field at the seam of the closing, which MLPs cannot model easily. This is a consequence of the fact that coordinate-based MLPs with positional encoding perform interpolation with a band-limited kernel, when viewed through the lens of neural tangent kernels [202].

Modulation-based NeRFs Modulation-based (or latent-conditioned) NeRF variants adopt a substantially different approach for handling dynamic scenes. Instead of modeling the scene as a single NeRF that is warped to explain individual images, these techniques provide additional information (e.g., the image's timestamp or index) as input to the MLP, parameterizing the radiance field of the scene. As such, these techniques are capable of modeling any deformation, topological change, or even complex phenomena such as fire. However, because these modulated NeRFs may have completely different radiance and density across input images, these techniques result in a severely under-constrained problem and allows for trivial, non-plausible solutions. This issue can be addressed by providing additional supervision such as depth and optical flow (as in Video-NeRF [229] and NSFF [119]) or by using a multi-view input captured from 7 synchronized cameras (as in DyNeRF [114]). NeRF in the Wild also uses a similar modulated approach (albeit for a different task) in which latent codes are optimized and provided as input to an MLP, which also introduces ambiguities that are addressed by allowing those codes to only modify radiance but not density [136].

2.4 Casual 3D Capture

In this section I provide an overview of how methods address casual 3D capture as an application. Broadly speaking, casual 3D capture strives to make 3D reconstruction easy

for the average user. This involves relaxing assumptions to allow for more casual capturing regimes (e.g. just waving your phone around) and eliminating barriers to entry such as the need for complicated and expensive hardware.

Single-shot Capture There is a topic of research that takes the term "casual" to the extreme. Instant 3D photography [74] and Stereo Magnification [245] focus on the problem of creating 3D experiences using a single capture. Both of these works attempt to use the rise of dual-lens mobile phone cameras to compute depth maps and lift a photo into 3D. Instant 3D constructs a mesh and does simple hole filling to make the resulting renderings look compelling while Stereo Magnification trains a neural network that can fill in the holes. As simple as the input is, these methods cannot be used to render viewpoints that are far from the capture, and must be able to hallucinate details not captured in the single photo making it a very under-constrained problem.

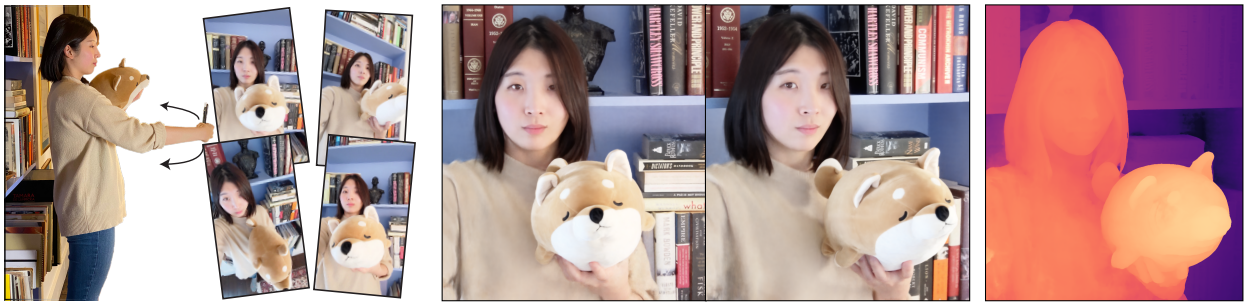
Dense Capture The line of research that is closest to my work is those that instruct the user to capture the scene as completely as possible. The difference between this and early light field based methods is that we do not want to require a special capture setup and allow the user to follow rough guidelines. Casual 3D Photography [75] creates a textured, normal-mapped mesh representation that can be rendered using standard 3D rasterization pipelines from images captured casually using either a mobile phone or DSLR. It creates partial panoramas using traditional SfM and MVS methods and then stitches them into a single panorama depth depth and color. These can then be fused to create a 3D photo that can be rendered from novel views near the center of the scene. Because it uses existing SfM and MVS methods, it inherits their limitations such as the inability to handle reflective of transculent surfaces, thin structures, and dynamic scenes. Unlike Deep Blending [76], Casual 3D Photography does not perform any blending and thus can show visible color discontinuities at the seams of the stitched partial panoramas. LLFF [142] observed that existing methods have no guidelines on how densely users should capture a given scene,

and devise a scheme to compute and prescribe a sample rate that can be communicated to the user. LLFF uses local MPI representation and blends between the resulting light fields (hence the name). Most recently, Neural Radiance Fields (NeRF) [143] have emerged as the front-runner of dense capture methods. It supports the same kinds of casual captures as Casual 3D Photography and LLFF but is much more robust in low-data regimes. In addition, it can handle reflective and translucent surface as well and thin structures with unprecedented quality. The two works presented in this works both build on NeRF.

Performance Capture The methods introduced in the previous paragraph all focus on capturing a static scene. The world is of course not static, and the most interesting subjects — people — cannot stand still even when they try. The task of *performance capture* can be thought of as equivalent to dense capture, except we are also interested in the temporal domain. Moving scenes are difficult to capture with a monocular camera since the scene is evolving as the camera moves. Therefore methods that attempt to reconstruct moving scenes such as High Quality Streamable Video [43], LookinGood [135], and Neural Volumes [127] require multiple, synchronized cameras in a studio setting to capture the scene simultaneously. Our goal in casual 3D capture is to avoid the use of specialized equipment. The next chapter describes my work on reconstructing moving scenes with a monocular camera.

Chapter 3

CASUAL CAPTURE OF DEFORMABLE SCENES



(a) casual capture (b) input images (c) nerfie novel views (d) novel view depth

Figure 3.1: We reconstruct photo-realistic *nerfies* from users casually waving a mobile phone (a). Our system uses selfies (b) to produce a free-viewpoint representation with accurate renders (c) and geometry (d).

Modeling real-world objects such as people with hand-held cameras is often challenging due to 1) nonrigidity – the tendency for things to move, and 2) challenging materials like hair, glasses, and earrings that violate assumptions used in most reconstruction methods. In this chapter, we introduce an approach to address both of these challenges, by generalizing Neural Radiance Fields (NeRF) [143] to model shape deformations. Our technique recovers high fidelity 3D reconstructions from short videos, providing free-viewpoint visualizations while accurately capturing hair, glasses, and other complex, view-dependent materials, as shown in Figure 4.1. A special case of particular interest is capturing a 3D self-portrait – we call such casual 3D selfie reconstructions *nerfies*.

Rather than represent shape explicitly, NeRF [143] uses a neural network to encode color and density as a function of location and viewing angle, and generates novel views using volume rendering. Their approach produces 3D visualizations of unprecedented quality, faithfully representing thin structures, semi-transparent materials, and view-dependent effects. To model non-rigidly deforming scenes, we generalize NeRF by introducing an ad-

ditional component: A canonical NeRF model serves as a template for all the observations, supplemented by a deformation field for each observation that warps 3D points in the frame of reference of an observation into the frame of reference of the canonical model. We represent this deformation field as a multi-layer perceptron (MLP), similar to the radiance field in NeRF. This deformation field is conditioned on a per-image learned latent code, allowing it to vary between observations.

Without constraints, the deformation fields are prone to distortions and over-fitting. We employ a similar approach to the elastic energy formulations that have seen success for mesh fitting [20, 32, 194, 196]. However, our volumetric deformation field formulation greatly simplifies such regularization, because we can easily compute the Jacobian of the deformation field through automatic differentiation, and directly regularize its singular values.

To robustly optimize the deformation field, we propose a novel coarse-to-fine optimization scheme that modulates the components of the input positional encoding of the deformation field network by frequency. By zeroing out the high frequencies at the start of optimization, the network is limited to learn smooth deformations, which are later refined as higher frequencies are introduced into the optimization.

For evaluation, we capture image sequences from a rig of two synchronized, rigidly attached, calibrated cameras, and use the reconstruction from one camera to predict views from the other. We plan to release the code and data.

In summary, our contributions are: ① an extension to NeRF to handle non-rigidly deforming objects that optimizes a deformation field per observation; ② rigidity priors suitable for deformation fields defined by neural networks; ③ a coarse-to-fine regularization approach that modulates the capacity of the deformation field to model high frequencies during optimization; ④ a system to reconstruct free-viewpoint selfies from casual mobile phone captures.

3.1 Related Work

Non-Rigid Reconstruction: Non-rigid reconstruction decomposes a scene into a geometric model and a deformation model that deforms the geometric model for each observation. Earlier works focused on sparse representations such as keypoints projected onto 2D images [24, 207], making the problem highly ambiguous. Multi-view captures [43, 53] simplify the problem to one of registering and fusing 3D scans [113]. DynamicFusion [149] uses a single RGBD camera moving in space, solving jointly for a canonical model, a deformation, and camera pose. More recently, learning-based methods have been used to find correspondences useful for non-rigid reconstruction [21, 177]. Unlike prior work, our method does not require depth nor multi-view capture systems and works on monocular RGB inputs. Most similar to our work, Neural Volumes [127] learns a 3D representation of a deformable scene using a voxel grid and warp field regressed from a 3D CNN. However, their method requires dozens of synchronized cameras and our evaluation shows that it does not extend to sequences captured from a single camera. Yoon *et al.* [237] reconstruct dynamic scenes from moving camera trajectories, but their method relies on strong semantic priors, in the form of monocular depth estimation, which are combined with multi-view cues. OFlow [152] solves for temporal flow-fields using ODEs, and thus requires temporal information. ShapeFlow [96] learns 3D shapes a divergence-free deformations of a learned template. Instead, we propose an elastic energy regularization.

Domain-Specific Modeling: Many reconstruction methods use domain-specific knowledge to model the shape and appearance of categories with limited topological variation, such as faces [15, 18, 19], human bodies [128, 233], and animals [28, 249]. Although some methods show impressive results in monocular face reconstruction from color and RGBD cameras [248], such models often lack detail (e.g., hair), or do not model certain aspects of a category (e.g., eyewear or garments). Recently, image translation networks have been applied to improve the realism of composited facial edits [58, 102]. In contrast, our work does not rely on domain-specific knowledge, enabling us to model the whole scene, including

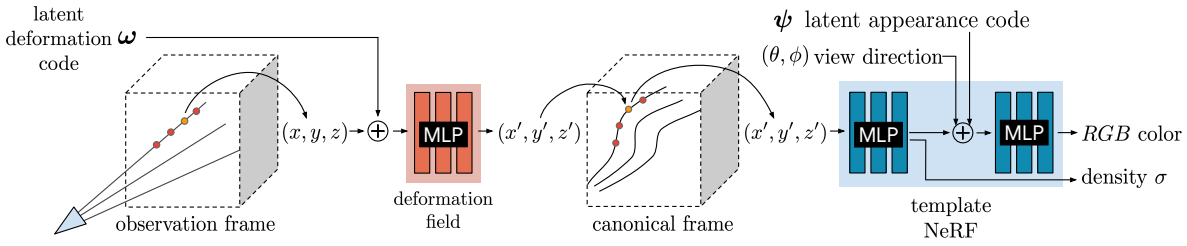


Figure 3.2: We associate a latent deformation code ($\boldsymbol{\omega}$) and an appearance code ($\boldsymbol{\psi}$) to each image. We trace the camera rays in the observation frame and transform samples along the ray to the canonical frame using a deformation field encoded as an MLP that is conditioned on the deformation code $\boldsymbol{\omega}$. We query the template NeRF [143] using the transformed sample (x', y', z'), viewing direction (θ, ϕ) and appearance code $\boldsymbol{\psi}$ as inputs to the MLP and integrate samples along the ray following NeRF.

eyeglasses and hair for human subjects.

Coordinate-based Models: Our method builds on the recent success of coordinate-based models, which encode a spatial field in the weights of a multilayer perceptron (MLP) and require significantly less memory compared to discrete representations. These methods have been used to represent shapes [38, 140, 156] and scenes [143, 191]. Of particular interest are NeRFs [143], that use periodic positional encoding layers [192, 202] to increase resolution, and whose formulation has be extended to handle different lighting conditions [14, 136], transient objects [136], large scenes [123, 241] and to model object categories [180]. Our work extends NeRFs to handle non-rigid scenes.

Similar Work: Two works [168, 209] also propose to represent deformable scenes using a translation field in conjunction with a template. This is similar to our framework with the following differences: ① we condition the deformation with a per-example latent [17] instead of time [168]; ② propose an as-rigid-as-possible regularization of the deformation field while NR-NeRF [209] penalizes the divergence of the translation field; ③ propose a coarse-to-fine regularization to prevent getting stuck in local minima; and ④ propose an improved SE(3) parameterization of the deformation field. Other concurrent works [118, 228] reconstruct space-time videos by recovering time-varying NeRFs while leveraging external supervision such as monocular depth estimation and flow-estimation to resolve ambiguities.

3.2 Deformable Neural Radiance Fields

Here we describe our method for modeling non-rigidly deforming scenes given a set of casually captured images of the scene. We decompose a non-rigidly deforming scene into a template volume represented as a neural radiance field (NeRF) [143] (§4.3.1) and a per-observation deformation field (§3.2.2) that associates a point in observation coordinates to a point on the template (overview in Figure 4.4). The deformation field is our key extension to NeRF and allows us to represent moving subjects. Jointly optimizing a NeRF together with a deformation field leads to an under-constrained optimization problem. We therefore introduce an elastic regularization on the deformation (§3.2.3), a background regularization (§3.2.4), and a continuous, coarse-to-fine annealing technique that avoids bad local minima (§3.2.5).

3.2.1 Neural Radiance Fields

A neural radiance field (NeRF) is a continuous, volumetric representation. It is a function $F : (\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma)$ which maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\phi, \theta)$ to a color $\mathbf{c} = (r, g, b)$ and density σ. In practice, NeRF maps the inputs $\mathbf{x}$ and $\mathbf{d}$ using a sinusoidal positional encoding $\gamma : \mathbb{R}^3 \rightarrow \mathbb{R}^{3+6m}$ defined as $\gamma(\mathbf{x}) = \big(\mathbf{x}, \cdots, \sin\left(2^k \pi \mathbf{x}\right), \cos\left(2^k \pi \mathbf{x}\right), \cdots\big)$, where m is a hyper-parameter that controls the total number of frequency bands

and $k \in \{0, \ldots, m-1\}$. This function projects a coordinate vector $\mathbf{x} \in \mathbb{R}^3$ to a high dimensional space using a set of sine and cosine functions of increasing frequencies. This allows the MLP to model high-frequency signals in low-frequency domains as shown in [202]. Coupled with volume rendering techniques, NeRFs can represent scenes with photo-realistic quality. We build upon NeRF to tackle the problem of capturing deformable scenes.

Similar to NeRF-W [136], we also provide an appearance latent code $\boldsymbol{\psi}_i$ for each observed frame $i \in \{1, \ldots, n\}$ that modulates the color output to handle appearance variations between input frames, e.g., exposure and white balance.

The NeRF training procedure relies on the fact that given a 3D scene, two intersecting

rays from two different cameras should yield the same color. Disregarding specular reflection and transmission, this assumption is true for all static scenes. Unfortunately, many scenes are not completely static; e.g., it is hard for people to stay completely still when posing for a photo, or worse, when waving a phone when capturing themselves in a selfie video.

3.2.2 Neural Deformation Fields

With the understanding of this limitation, we extend NeRF to allow the reconstruction of non-rigidly deforming scenes. Instead of directly casting rays through a NeRF, we use it as a canonical template of the scene. This template contains the relative structure and appearance of the scene while a rendering will use a non-rigidly deformed version of the template (see Figure 3.4 for an example). DynamicFusion [149] and Neural Volumes [127] also model a template and a per-frame deformation, but the deformation is defined on mesh points and on a voxel grid respectively, whereas we model it as a continuous function using an MLP.

We employ an observation-to-canonical deformation for every frame $i \in \{1, \ldots, n\}$, where n is the number of observed frames. This defines a mapping $T_i : \mathbf{x} \rightarrow \mathbf{x}'$ that maps all observation-space coordinates $\mathbf{x}$ to a canonical-space coordinate $\mathbf{x}'$. We model the deformation fields for all time steps using a mapping $T : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{x}'$, which is conditioned on a per-frame learned latent deformation code $\boldsymbol{\omega}_i$. Each latent code encodes the state of the scene in frame i. Given a canonical-space radiance field F and a observation-to-canonical mapping T, the observation-space radiance field can be evaluated as:

$$G(\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i, \boldsymbol{\omega}_i) = F\left(T(\mathbf{x}, \boldsymbol{\omega}_i), \mathbf{d}, \boldsymbol{\psi}_i\right) . \tag{3.1}$$

When rendering, we simply cast rays and sample points in the observation frame and then use the deformation field to map the sampled points to the template, see Figure 4.4.

A simple model of deformation is a displacement field $V : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{t}$, defining the transformation as $T(\mathbf{x}, \boldsymbol{\omega}_i) = \mathbf{x} + V(\mathbf{x}, \boldsymbol{\omega}_i)$. This formulation is sufficient for all continuous deformations; however, rotating a group of points with a translation field requires a different

translation for each point, making it difficult to rotate regions of the scene simultaneously. We therefore formulate the deformation using a dense SE(3) field $W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \text{SE}(3)$. An SE(3) transform encodes rigid motion, allowing us to rotate a set of distant points with the same parameters.

We encode a rigid transform as a screw axis [133] $\mathcal{S} = (\mathbf{r}; \mathbf{v}) \in \mathbb{R}^6$. Note that $\mathbf{r} \in \mathfrak{so}(3)$ encodes a rotation where $\hat{\mathbf{r}} = \mathbf{r}/\|\mathbf{r}\|$ is the axis of rotation and $\theta = \|\mathbf{r}\|$ is the angle of rotation. The exponential of $\mathbf{r}$ (also known as Rodrigues' formula [173]) yields a rotation matrix $e^{\mathbf{r}} \in \text{SO}(3)$:

$$e^{\mathbf{r}} \equiv e^{[\mathbf{r}]_\times} = \mathbf{I} + \frac{\sin\theta}{\theta}[\mathbf{r}]_\times + \frac{1-\cos\theta}{\theta^2}[\mathbf{r}]_\times^2 \,, \tag{3.2}$$

where $[\mathbf{x}]_\times$ denotes the cross-product matrix of a vector $\mathbf{x}$ which is a skew-symmetric matrix also known as the cross-product matrix of a vector $\mathbf{x}$ since given two 3-vectors $\mathbf{a}$ and $\mathbf{b}$, $[\mathbf{a}]_\times \mathbf{b}$ gives the cross product $\mathbf{a} \times \mathbf{b}$.

$$[\mathbf{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix} \,. \tag{3.3}$$

Similarly, the translation encoded by the screw motion $\mathcal{S}$ can be recovered as $\mathbf{p} = \mathbf{G}\mathbf{v}$ where

$$\mathbf{G} = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\mathbf{r}]_\times + \frac{\theta - \sin\theta}{\theta^3}[\mathbf{r}]_\times^2 \,. \tag{3.4}$$

Combining these formulas and using the exponential map, we get the transformed point as $\mathbf{x}' = e^{\mathcal{S}}\mathbf{x} = e^{\mathbf{r}}\mathbf{x} + \mathbf{p}$.

As mentioned before, we encode the transformation field in an MLP $W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow (\mathbf{r}, \mathbf{v})$ using a NeRF-like architecture, and represent the transformation of every frame i by conditioning on a latent code $\boldsymbol{\omega}_i$. We optimize the latent code through an embedding layer [17]. Like with the template, we map the input $\mathbf{x}$ using positional encoding γ_α (see §3.2.5). An important property of the $\mathfrak{se}(3)$ representation is that $e^{\mathcal{S}}$ is the identity when $\mathcal{S} = \mathbf{0}$. We therefore initialize the weights of the last layer of the MLP from $\mathcal{U}(-10^{-5}, 10^{-5})$ to initialize the deformation near the identity.

Figure 3.4: Visualizations of the recovered 3D model in the observation and canonical frames of reference, with insets showing orthographic views in the forward and left directions. Note the right-to-left and front-to-back displacements between the observation and canonical model, which are modeled by the deformation field for this observation.

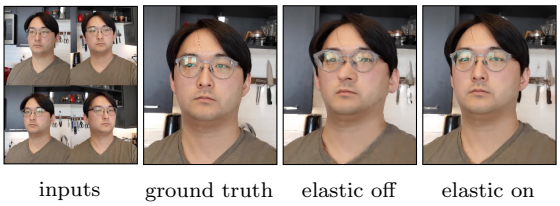


Figure 3.5: Our elastic regularization helps when the scene is under-constrained. This capture only contains 20 input images with the cameras biased towards one side of the face resulting in an under constrained problem. Elastic regularization helps resolve the ambiguity and leads to less distortion.

Why does an SE(3) field work better? Consider the example in Figure 3.3 where a star has been rotated counter-clockwise along its center. Now consider what transformation would be required at every point on the star to encode this rotation. With a translation field, points towards the center (e.g., $\mathbf{t_2}$) need translations of small magnitude while points towards the outside (e.g., $\mathbf{t_1}$) need translations of larger magnitude. Every point on the star requires a different parameter to encode a simple rotation. On the otherhand, with a rotation, every point on the star can be parameterized by a single angle which is the angle of rotation $\theta = \theta_1 = \theta_2$. This makes optimization much easier since the deformation field MLP only needs to predict a single parameter across space.

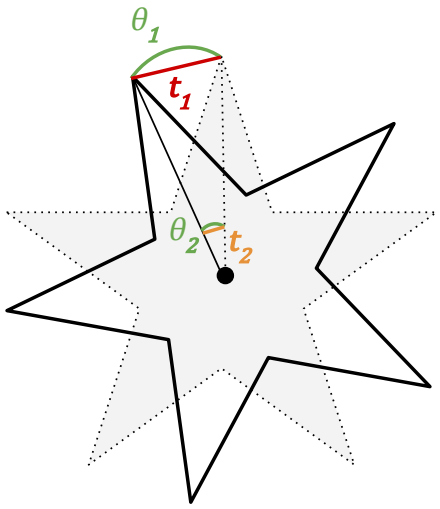


Figure 3.3: Here we illustrate why a rigid transformation field works better than a translation field with a simple toy example where a star is rotated counter-clockwise around its center. A translation field requires different parameters for every point to encode the rotation (e.g., $\|\mathbf{t_1}\| \gg \|\mathbf{t_2}\|$) whereas a rotation field only needs a single parameter to encode the rotation (e.g., $\theta_1 = \theta_2$).

3.2.3 Elastic Regularization

The deformation field adds ambiguities that make optimization more challenging. For example, an object moving backwards is visually equivalent to it shrinking in size, with many solutions in between. These ambiguities lead to under-constrained optimization problems which yield implausible results and artifacts (see Figure 3.7). It is therefore crucial to introduce priors that lead to a more plausible solution.

It is common in geometry processing and physics simulation to model non-rigid deformations using elastic energies measuring the deviation of local deformations from a rigid motion [20, 32, 194, 196]. In the vision community, these energies have been extensively used for the reconstruction and tracking of non-rigid scenes and objects [53, 149, 247] making them good candidates for our approach. While they have been most commonly used for discretized surfaces, e.g., meshes, we can apply a similar concept in the context of our continuous deformation field.

Elastic Energy: For a fixed latent code $\boldsymbol{\omega}_i$, our deformation field T is a non-linear mapping from observation-coordinates in $\mathbb{R}^3$ to canonical coordinates in $\mathbb{R}^3$. The Jacobian $\mathbf{J}_T(\mathbf{x})$ of this mapping at a point $\mathbf{x} \in \mathbb{R}^3$ describes the best linear approximation of the transformation at that point. We can therefore control the local behavior of the deformation through $\mathbf{J}_T$ [186]. Note that unlike other approaches using discretized surfaces, our continuous formulation allows us to directly compute $\mathbf{J}_T$ through automatic differentiation of the MLP. There are several ways to penalize the deviation of the Jacobian $\mathbf{J}_T$ from a rigid transformation. Considering the singular-value decomposition of the Jacobian $\mathbf{J}_T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, multiple approaches [20, 32] penalize the deviation from the closest rotation as $\|\mathbf{J}_T - \mathbf{R}\|_F^2$, where $\mathbf{R} = \mathbf{V}\mathbf{U}^T$ and $\|\cdot\|_F$ is the Frobenius norm. We opt to directly work with the singular values of $\mathbf{J}_T$ and measure its deviation from the identity. The log of the singular values gives equal weight to a contraction and expansion of the same factor, and we found it to perform better. We therefore penalize the deviation of the log singular values from zero:

$$L_{\text{elastic}}(\mathbf{x}) = \|\log \boldsymbol{\Sigma} - \log \mathbf{I}\|_F^2 = \|\log \boldsymbol{\Sigma}\|_F^2 \,, \tag{3.5}$$

where log here is the matrix logarithm.

Equivalency to closest rotation Elastic energies are often implemented as the deviation of the Jacobian $\mathbf{J}$ from the closest rotation $\mathbf{R}$: $\|\mathbf{J}-\mathbf{R}\|_F$ [p45]. Let $\mathbf{J}=\mathbf{U}\mathbf{D}\mathbf{V}^T$ be the SVD of $\mathbf{J}$, then $\mathbf{R}=\mathbf{U}\mathbf{M}\mathbf{V}^T$ where $\mathbf{M}=\mathrm{diag}\Big(\Big)(1,\ldots,1,\det(\mathbf{U}\mathbf{V}^T)\Big)$. It follows that:

$$\begin{aligned}
\|\mathbf{J}-\mathbf{R}\|_F &= \left\|\mathbf{U}\mathbf{D}\mathbf{V}^T-\mathbf{U}\mathbf{M}\mathbf{V}^T\right\|_F && (3.6)\\
&= \left\|\mathbf{U}(\mathbf{D}-\mathbf{M})\mathbf{V}^T\right\|_F && (3.7)\\
&= \sqrt{\mathrm{tr}\left(\mathbf{U}(\mathbf{D}-\mathbf{M})\mathbf{V}^T\mathbf{V}(\mathbf{D}-\mathbf{M})\mathbf{U}^T\right)} && (3.8)\\
&= \sqrt{\mathrm{tr}\left(\mathbf{U}(\mathbf{D}-\mathbf{M})^2\mathbf{U}^T\right)} && (3.9)\\
&= \sqrt{\mathrm{tr}\left((\mathbf{D}-\mathbf{M})^2\right)} && (3.10)\\
&= \sqrt{\sum_j(\sigma_j-m_j)^2}\,, && (3.11)
\end{aligned}$$

where m_j is the jth diagonal of $\mathbf{M}$ and σ_j is the jth singular value of $\mathbf{J}$. This is equivalent to penalizing the deviation of the singular values of $\mathbf{J}$ from 1. The $\mathbf{M}$ matrix factors in reflections as negative singular values rather a reflection in $\mathbf{U}$ or $\mathbf{V}$. Because this formulation penalizes expansions more than contractions of the same factor, we penalize the log of the singular values directly.

Robustness: Although humans are mostly rigid, there are some movements which can break our assumption of local rigidity, e.g., facial expressions which locally stretch and compress our skin. We therefore remap the elastic energy defined above using a robust loss:

$$\begin{aligned}
L_{\text{elastic-r}}(\mathbf{x}) &= \rho\left(\|\log\boldsymbol{\Sigma}\|_F, c\right), && (3.12)\\
\rho(x,c) &= \frac{2(x/c)^2}{(x/c)^2+4}\,. && (3.13)
\end{aligned}$$

where $\rho(\cdot)$ is the Geman-McClure robust error function [63] parameterized with hyperparameter $c=0.03$ as per Barron [8]. This robust error function causes the gradients of the loss to fall off to zero for large values of the argument, thereby reducing the influence of outliers during training.

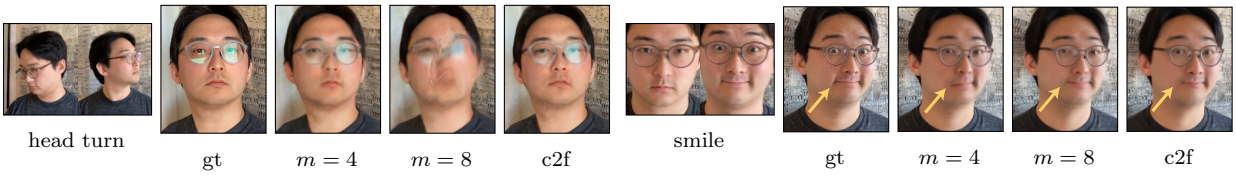


Figure 3.6: In this capture, the subject rotates their head (top) and smiles (bottom). With $m = 4$ positional encoding frequencies, the deformation model does not capture the smile, while it fails to rotate the head with $m = 8$ frequencies. With coarse-to-fine regularization (c2f) the model captures both.

Weighting: We allow the deformation field to behave freely in empty space, since the subject moving relative to the background requires a non-rigid deformation somewhere in space. We therefore weight the elastic penalty at each sample along the ray by its contribution to the rendered view, *i.e.* w_i in Eqn. 5 of NeRF [143].

3.2.4 Background Regularization

The deformation field is unconstrained and therefore everything is free to move around. We optionally add a regularization term which prevents the background from moving. Given a set of 3D points in the scene which we know should be static, we can penalize any deformations at these points. For example, camera registration using structure from motion produces a set of 3D feature points that behave rigidly across at least some set of observations. Given these static 3D points $\{\mathbf{x}_1 \ldots, \mathbf{x}_K\}$, we penalize movement as:

$$L_{\text{bg}} = \frac{1}{K} \sum_{k=1}^{K} \left\| T(\mathbf{x}_k) - \mathbf{x}_k \right\|_2 . \tag{3.14}$$

In addition to keeping the background points from moving, this regularization also has the benefit of aligning the observation coordinate frame to the canonical coordinate frame.

3.2.5 Coarse-to-Fine Deformation Regularization

A common trade-off that arises during registration and flow estimation is the choice between modeling minute versus large motions, that can lead to overly smooth results or incorrect registration (local minima). Coarse-to-fine strategies circumvent the issue by first solving

Figure 3.7: Novel views synthesized by linearly interpolating the deformation latent codes of two frames of the BADMINTON (left and right) show a smooth racquet motion.

the problem in low-resolution, where motion is small, and iteratively upscaling the solution and refining it [131]. We observe that our deformation model suffers from similar issues, and propose a coarse-to-fine regularization to mitigate them.

Recall the positional encoding parameter m introduced in §4.3.1 that controls the number of frequency bands used in the encoding. Tancik *et al.* [202] show that controls it the smoothness of the network: a low value of m results in a low-frequency bias (low resolution) while a higher value of m results in a higher-frequency bias (high resolution).

Consider a motion like in Figure 3.6, where a subject rotates their head and smiles. With a small m, the model cannot capture the minute motion of the smile; conversely, with a larger m, the model fails to correctly rotate the head because the template overfits to an underoptimized deformation field. To overcome this, we propose a coarse-to-fine approach that starts with a low-frequency bias and ends with a high-frequency bias.

Tancik *et al.* [202] show that positional encoding can be interpreted in terms of the Neural Tangent Kernel (NTK) [92] of NeRF's MLP: a stationary interpolating kernel where m controls a tunable "bandwidth" of that kernel. A small number of frequencies induces a wide kernel which causes under-fitting of the data, while a large number of frequencies induces a narrow kernel causing over-fitting. With this in mind, we propose a method to smoothly anneal the bandwidth of the NTK by introducing a parameter α that windows the frequency bands of the positional encoding, akin to how coarse-to-fine optimization schemes solve for coarse solutions that are subsequently refined at higher resolutions. We define the

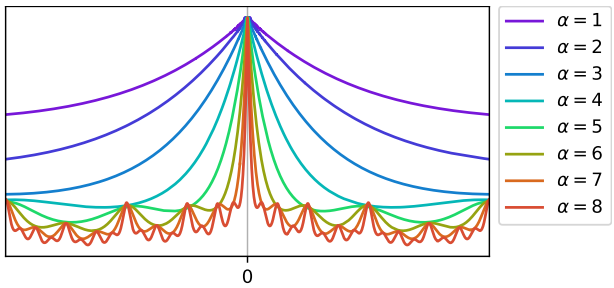


Figure 3.8: Visualizations of the neural-tangent kernel (NTK) [92] of our annealed positional encoding for different values of α. Our coarse-to-fine optimization scheme works by easing in the influence of each positional encoding frequency through a parameter α. This has the effect of shrinking the bandwidth of the NTK corresponding to the deformation MLP as α is increased, thereby allowing higher frequency deformations.

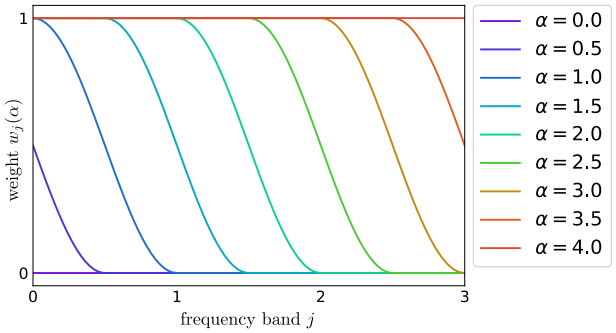


Figure 3.9: A visualization of the window function $w_j(\alpha)$ for the annealed positional encoding. We show an example with a maximum number of frequency bands of $m = 4$ where $j \in \{0, \ldots, m-1\}$. $\alpha = 0$ sets the weight of all frequency bands to zero leaving only the identity mapping, while an $\alpha = 4$ sets the weight of all frequency bands to one. Increasing the value of α is equivalent to sliding the window to the right across the frequency bands.

weight for each frequency band j as:

$$w_j(\alpha) = \frac{(1 - \cos(\pi \operatorname{clamp}(\alpha - j, 0, 1))}{2}, \tag{3.15}$$

where linearly annealing the parameter $\alpha \in [0, m]$ can be interpreted as sliding a truncated Hann window (where the left side is clamped to 1 and the right side is clamped to 0) across the frequency bands. The positional encoding is then defined as $\gamma_\alpha(\mathbf{x}) = \big(\mathbf{x}, \cdots, w_k(\alpha)\sin(2^k\pi\mathbf{x}), w_k(\alpha)\cos(2^k\pi\mathbf{x}), \cdots\big)$. During training, we set $\alpha(t) = \frac{mt}{N}$ where t is the current training iteration, and N is a hyper-parameter for when α should reach the maximum number of frequencies m.

We visualize our windowing function for different values of α in Figure 3.9.

NTK: We also show a visualization of the neural tangent kernel (NTK) induced by our annealed positional encoding in Figure 3.8. This figure shows the normalized NTK for an 8 layer MLP of width 256. Note how the bandwidth of the interpolation kernel gets narrower as the value of α increases.

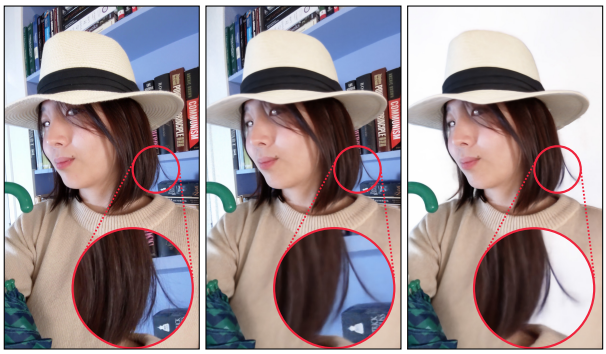


Figure 3.10: Our method recovers thin hair strands. By adjusting the camera's far plane, we can render the subject against a flat white background.

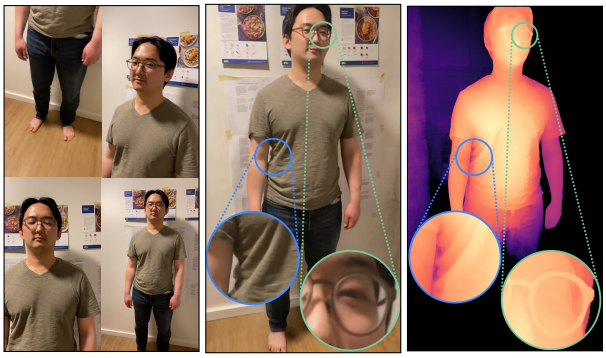


Figure 3.11: Our method reconstructs full body scenes captured by a second user with high quality details.

3.2.6 Nerfies: Casual Free-Viewpoint Selfies

So far we have presented a generic method of reconstructing non-rigidly deforming scenes. We now present a key application of our system – reconstructing high quality models of human subjects from casually captured selfies, which we dub *"nerfies"*. Our system takes as input a sequence of selfie photos or a selfie video in which the user is standing mostly still. Users are instructed to wave the camera around their face, covering viewpoints within a 45° cone. We observe that 20 second captures are sufficient. In our method, we assume that the subject stands against a static background to enable a consistent geometric registration of the cameras. We filter blurry frames using the variance of the Laplacian [163], keeping about 600 frames per capture.

Camera Registration: We seek a registration of the cameras with respect to the static background. We use COLMAP [178] to compute pose for each image and camera intrinsics. This step assumes that enough features are present in the background to register the sequence.

Foreground Segmentation: In some cases, SfM will match features on the moving subject, causing significant misalignment in the background. This is problematic in video captures with correlated frames. In those cases, we found it helpful to discard image features on the

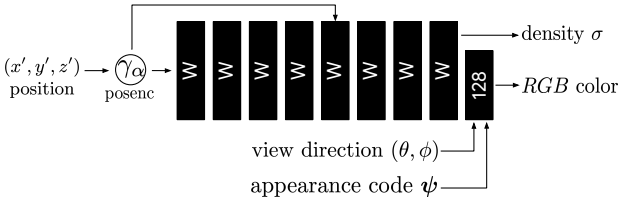


Figure 3.12: A diagram of the canonical NeRF network. Our network is identical to the original NeRF MLP, except we provide an appearance latent code $\boldsymbol{\psi}$ along with the view direction to allow modulating the appearance as in the NeRF-A model of [136].

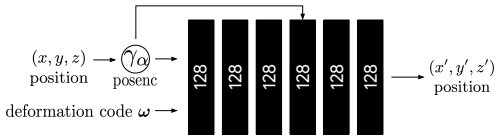


Figure 3.13: A diagram of our deformation network. The deformation network takes a position encoded position $\gamma_\alpha(\mathbf{x})$ using our coarse-to-fine annealing parameterized by α, along with a deformation code $\boldsymbol{\omega}$ and outputs a deformed position $\mathbf{x}'$. The architecture is identical for all of our experiments.

subject, which can be detected using a foreground segmentation network.

3.3 Experiments

3.3.1 Implementation Details

Our NeRF template implementation closely follows the original [143], except we use a Softplus activation $\ln(1+e^x)$ for the density. We use a deformation network with depth 6, hidden size 128, and a skip connection at the 4th layer. We show diagrams of our architecture in Figure 3.12 and Figure 3.13. We use 256 coarse and fine ray samples for full HD (1920×1080) models and half that for the half resolution models. We use 8 dimensions for the latent deformation and appearance codes. For coarse-to-fine optimization we use 6 frequency bands and linearly anneal α from 0 to 6 over 80K iterations. We use the same MSE photometric loss as in NeRF [143] and weight the losses as $L_{\text{total}} = L_{\text{rgb}} + \lambda L_{\text{elastic-r}} + \mu L_{\text{bg}}$ where we use $\lambda = \mu = 10^{-3}$ for all experiments except when mentioned. We train on 8 V100 GPUs for a week for full HD models, and for 16 hours for the half resolution models used for the comparisons in Tab. 3.1, Figure 3.15, and Figure 3.16.

3.3.2 Evaluation Dataset

In order to evaluate the quality of our reconstruction, we must be able to measure how faithfully we can recreate the scene from a viewpoint unseen during training. Since we are reconstructing non-rigidly deforming scenes, we cannot simply hold out views from an input

Table 3.1: Quantitative evaluation on validation captures against baselines and ablations of our system, we color code each row as **best**, **second best**, and **third best**. †denotes use of temporal information. Please see Sec. 4.4.2 for more details.

	Quasi-Static														Dynamic									
	Glasses (78 images)		Beanie (74 images)		Curls (57 images)		Kitchen (40 images)		Lamp (55 images)		Toby Sit (308 images)		Mean		Drinking (193 images)		Tail (238 images)		Badminton (356 images)		Broom (197 images)		Mean	
	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
NeRF [143]	18.1	.474	16.8	.583	14.4	.616	19.1	.434	17.4	.444	22.8	.463	18.1	.502	18.6	.397	23.0	.571	18.8	.392	21.0	.667	20.3	.506
NeRF + latent	19.5	.463	19.5	.535	17.3	.539	20.1	.403	18.9	.386	19.4	.385	19.1	.452	21.9	.233	24.9	.404	20.0	.308	21.9	.576	22.2	.380
Neural Volumes [127]	15.4	.616	15.7	.595	15.2	.588	16.2	.569	13.8	.533	13.7	.473	15.0	.562	16.2	.198	18.5	.559	13.1	.516	16.1	.544	16.0	.454
NSFF†	19.6	.407	21.5	.402	18.0	.432	21.4	.317	20.5	.239	26.9	.208	21.3	.334	27.7	.0803	30.6	.245	21.7	.205	28.2	.202	27.1	.183
$\gamma(t)$ + Trans† [118]	22.2	.354	20.8	.471	20.7	.426	22.5	.344	21.9	.283	25.3	.420	22.2	.383	23.7	.151	27.2	.391	22.9	.221	23.4	.627	24.3	.347
Ours ($\lambda = 0.01$)	23.4	.305	22.2	.391	24.6	.319	23.9	.280	23.6	.232	22.9	.159	23.4	.281	22.4	.0872	23.9	.161	22.4	.130	21.5	.245	22.5	.156
Ours ($\lambda = 0.001$)	24.2	.307	23.2	.391	24.9	.312	23.5	.279	23.7	.230	22.8	.174	23.7	.282	21.8	.0962	23.6	.175	22.1	.132	21.0	.270	22.1	.168
No elastic	23.1	.317	24.2	.382	24.1	.322	22.9	.290	23.7	.230	23.0	.257	23.5	.300	22.2	.0863	23.7	.174	22.0	.132	20.9	.287	22.2	.170
No coarse-to-fine	23.8	.312	21.9	.408	24.5	.321	24.0	.277	22.8	.242	22.7	.244	23.3	.301	22.3	.0960	24.3	.257	21.8	.151	21.9	.406	22.6	.228
No SE3	23.5	.314	21.9	.401	24.5	.317	23.7	.282	22.7	.235	22.9	.206	23.2	.293	22.4	.0867	23.5	.191	21.2	.156	20.9	.276	22.0	.177
Ours (base)	24.0	.319	20.9	.456	23.5	.345	22.4	.323	22.1	.254	22.7	.184	22.6	.314	22.6	.127	24.3	.298	21.1	.173	22.1	.503	22.5	.275
No BG Loss	22.3	.317	21.5	.395	20.1	.371	22.5	.290	20.3	.260	22.3	.145	21.5	.296	22.3	.0856	23.5	.210	20.4	.161	20.9	.330	21.8	.196

capture, as the structure of the scene will be slightly different in every image. We therefore build a simple multi-view data capture rig for the sole purpose of evaluation. We found the multi-view dataset of Yoon *et al.* [237] not representative of many capture scenarios, as it contains too few viewpoints (12) and exaggerated frame-to-frame motions due to temporal subsampling.

Our rig (Figure 3.14) is a pole with two Pixel 3's rigidly attached. We have two methods for data capture: (a) for selfies we use the front-facing camera and capture time-synchronized photos using the method of Ansari *et al.* [6], which achieves sub millisecond synchronization; or (b) we use the back-facing camera and record two videos which we manually synchronize based on the audio; we then subsample to 5 fps. We register the images using COLMAP [178] with rigid relative camera pose constraints. Sequences captured with (a) contain fewer frames (40˜78) but the focus, exposure, and time are precisely synchronized. Sequences captured with (b)

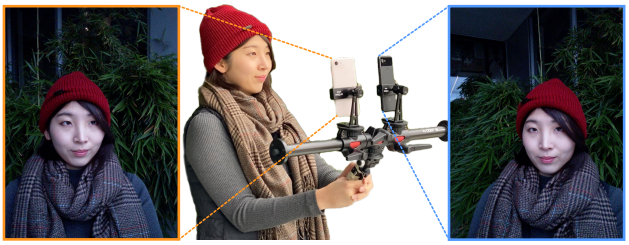


Figure 3.14: Validation rig used only for evaluation.

have denser samples in time (between 193 and 356 frames) but the synchronization is less precise and exposure and focus may vary between the cameras. We split each capture into a training set and a validation set. We alternate assigning the left view to the training set, and right to the validation, and vice versa. This avoids having regions of the scene that one camera has not seen.

Quasi-static scenes: We capture 5 human subjects using method (a), that attempt to stay as still as possible during capture, and a mostly still dog using method (b).

Dynamic scenes: We capture 4 dynamic scenes containing deliberate motions of a human subject, a dog wagging its tail, and two moving objects using method (b).

3.3.3 Evaluation

Here we provide quantitative and qualitative evaluations of our model. However, to best appreciate the quality of the reconstructed *nerfies*, we encourage the reader to watch the supplementary video that contains many example results.

Quantitative Evaluation: We compare against NeRF and a NeRF + latent baseline, where NeRF is conditioned on a per-image learned latent code [17] to modulate density and color. We also compare with a variant of our system similar to the concurrent work of D-NeRF [168], which conditions a translational deformation field with a position encoded time variable $\gamma(t)$ instead of a latent code ($\gamma(t)$+trans in Tab. 3.1). We also compare with the high quality model of Neural Volumes (NV) [127] using a single view as input to the encoder, and Neural Scene Flow Fields (NSFF) [118]. We do not evaluate the method of Yoon et al. [237] due to the lack of available code (note that NSFF outperforms it). NSFF and the $\gamma(t) + trans$ baseline use temporal information while other baselines and our method do not. NSFF also uses auxilliary supervision such as estimated flow and relative depth maps; we do not. Photometric differences between the two rig cameras may exist due to different exposure/white balance settings and camera response curves. We therefore swap the per-frame appearance code ψ_i for a per-camera $\{\psi_L, \psi_R\} \in \mathbb{R}^2$ instead for validation rig captures.

Tab. 3.1 reports LPIPS [242] and PSNR metrics for the unseen validation views. PSNR favors blurry images and is therefore not an ideal metric for dynamic scene reconstruction; we find that LPIPS is more representative of visual quality. See Figure 3.15 and Figure 3.16 for side-by-side images with associated PSNR/LPIPS metrics. Our method struggles with PSNR due to slight misalignments resulting from factors such as gauge ambiguity [138] while we outperform all baselines in terms of LPIPS for all sequences.

Ablation Study: We evaluate each of our contributions: SE(3) deformations, elastic regularization, background regularization, and coarse-to-fine optimization. We ablate them one at a time, and all at once (Ours (bare) in Tab. 3.1). As expected, a stronger elastic regularization ($\lambda = 0.01$) improves results for dynamic scenes compared to the baseline ($\lambda = 0.001$) while minimally impacting quasi-static scenes. Removing the elastic loss hurts performance for quasi-static scenes while having minimal effect on the dynamic scene; this may be due to the larger influence of other losses in the presence of larger motion. Elastic regularization fixes distortion artifacts when the scene is under-constrained (e.g., Figure 3.5). Disabling coarse-to-fine regularization mildly drops performance for quasi-static scenes while causing a significant drop for dynamic scenes. This is expected since large motions are a main source of local minima (e.g., Figure 3.6). Our SE(3) deformations also quantitatively outperform translational deformations. Background regularization helps PSNR by reducing shifts in static regions and removing it performs worse. Finally, removing all of our contributions performs the worst in terms of LPIPS.

Qualitative Results: We show results for the captures used in the quantitative evaluation in Figure 3.15 and Figure 3.16. Our method can reconstruct fine details such as strands of hair (e.g., in Curls of Tab. 3.1 and Figure 3.10), shirt wrinkles, and glasses (Figure 3.11). Our method works on general scenes beyond human subjects as shown in Figure 3.15 and Figure 3.16. In addition, we can create smooth animations by interpolating the deformation latent codes of any input state as shown in Figure 3.7.

Elastic Regularization: Figure 3.5 shows an example where the user only captured 20 images mostly from one side of their face, while their head tracked the camera. This results in

Figure 3.15: Comparisons of baselines and our method on quasi-static scenes. PSNR / LPIPS metrics on bottom right with best colored red.

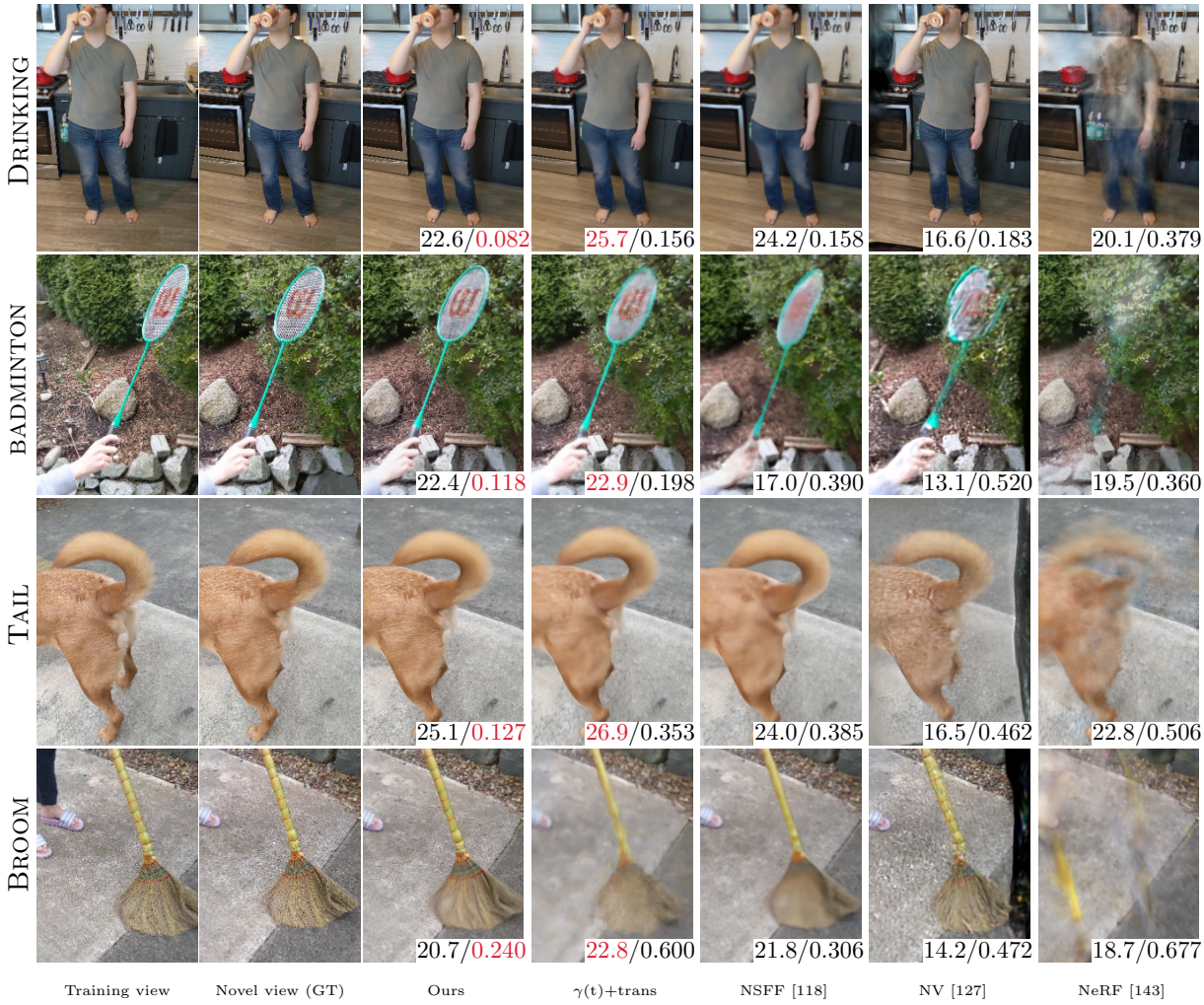


Figure 3.16: Comparisons of baselines and our method on dynamic scenes. PSNR / LPIPS metrics on bottom right with best colored red.

ambiguous geometry. Elastic regularization helps in such under-constrained cases, reducing distortion significantly.

Depth Visualizations: We visualize the quality of our reconstruction using depth renders of the density field. Unlike NeRF[143] that visualizes the expected ray termination distance, we use the *median* depth termination distance, which we found to be less biased by residual density in free space (see Fig. 3.10). We define it as the depth of the first sample with accumulated transmittance $T_i \geq 0.5$ (Eqn. 3 of NeRF [143]).

Figure 3.17: Our 2D toy dataset comprised of an image with a random translation, a random rotation, and a random non-linear distortion near the center. Astronaut photo by Robert Gibson (1984, Public Domain).

3.3.4 2D Deformation Experiments

Here we analyze the behavior of a deformation field in a 2D toy setting. In this 2D setting, a "scene" is comprised of a single image which is randomly translated, rotated, and non-linearly distorted near the center. We show the full dataset in Figure 3.17. Akin to our deformable NeRF setting, the task is to reconstruct each image by using a 2D deformation field which references a single template. The template is an MLP $F : (x, y) \rightarrow (r, g, b)$ which maps normalized image coordinates $x, y \in [-1, 1]$ to color values. The deformation field (i.e., 2D flow) is represented as an MLP $T : (x, y) \rightarrow (t_x, t_y)$ for a translation field or $T : (x, y) \rightarrow (\theta, p_x, p_y, t_x, t_y)$ for a rigid SE(2) transformation field. These are 2D analogs to the 3D translation field and SE(3) field described in Sec. 3.2.

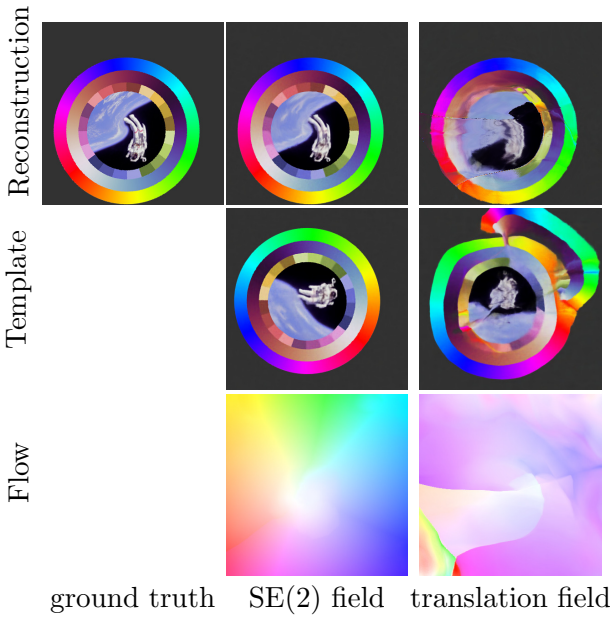


Figure 3.19: A comparison of our SE(2) field and a translation field. The translation parameterization has difficulty rotating groups of distant pixels whereas the rigid transformation successfully finds the correct orientation.

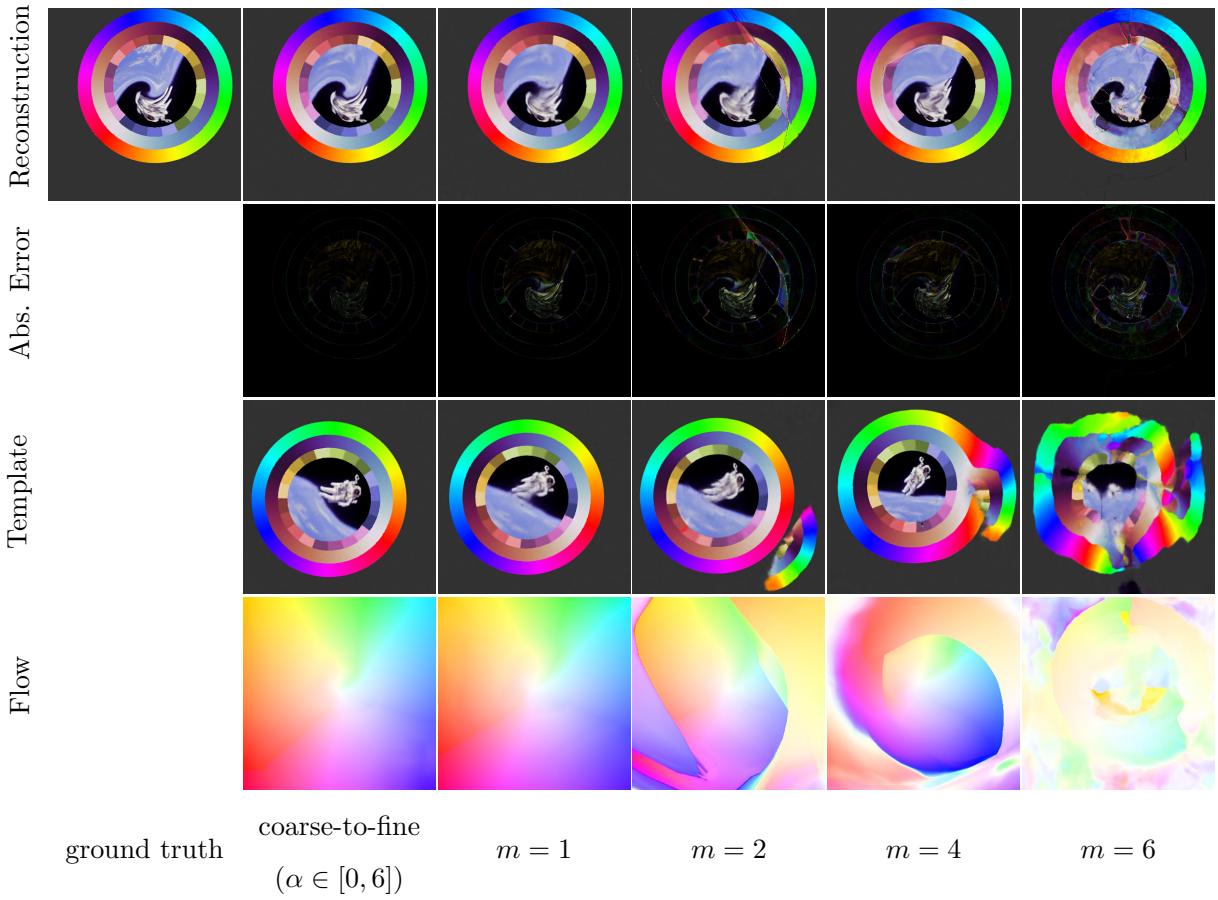


Figure 3.18: We show how the optimization changes depending on the number of frequency bands in the positional encoding of the deformation field. With 1 frequency, the model find the correct orientation of all images but is unable to model the high frequency distortion near the center. If we increase the frequencies then the templates overfits early and gets stuck in bad local minima. With our coarse-to-fine technique we are less prone to local minima while also modeling high frequency details.

Deformation Formulation: Figure 3.19 shows how an SE(2) rigid transformation field outperforms a translation field. An SE(2) field is able to faithfully reconstruct each image with a reasonable template and smooth deformation field. On the other hand, a translation field is not able to recover a reasonable template, and as a result the reconstruction has many artifacts and the flow field is messy.

Positional Encoding Frequencies: We show how changing the number of frequency bands changes the convergence behavior in Figure 3.18. With a small number of frequencies ($m = 1$) we are able to converge to the correct orientation in the template, but cannot fully

model the non-linear 'swirl' towards the center of the image. If we increase the number of frequencies ($m = 2 \dots 6$) then while we can reconstruct the high swirl better, we start introducing artifacts due to early overfitting of the template and deformation field. With our coarse-to-fine approach we are able to both get the correct orientation without artifacts and also model the swirl.

3.4 Limitations

Topological Variation: Our method struggles when the scene has motion which varies the topology of the scene. For example, when a person opens their mouth (as in Fig. 11 of the main text), the effective topology of their head changes. This is problematic for our method since we use a continuous deformation field parameterized by an MLP. In order to understand this, consider the mouth example and suppose that the template contains the person with their mouth open. Suppose that $\mathbf{x}_U$ and $\mathbf{x}_L$ are two adjacent points near the seam of the lips, and the $\mathbf{x}_U$ is on the upper lip and $\mathbf{x}_L$ is on the lower lip. It is then evident that a sharp discontinuity in the deformation is required to map both points to their appropriate positions on the template. Such a discontinuity is difficult for our continuous MLP to predict. We find that instead the optimization will often yield an incorrect but valid solution e.g., it will explain a closing mouth by protruding the lip and pulling it down as in Figure 11 of the paper.

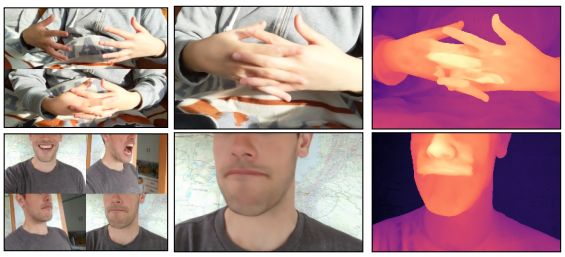
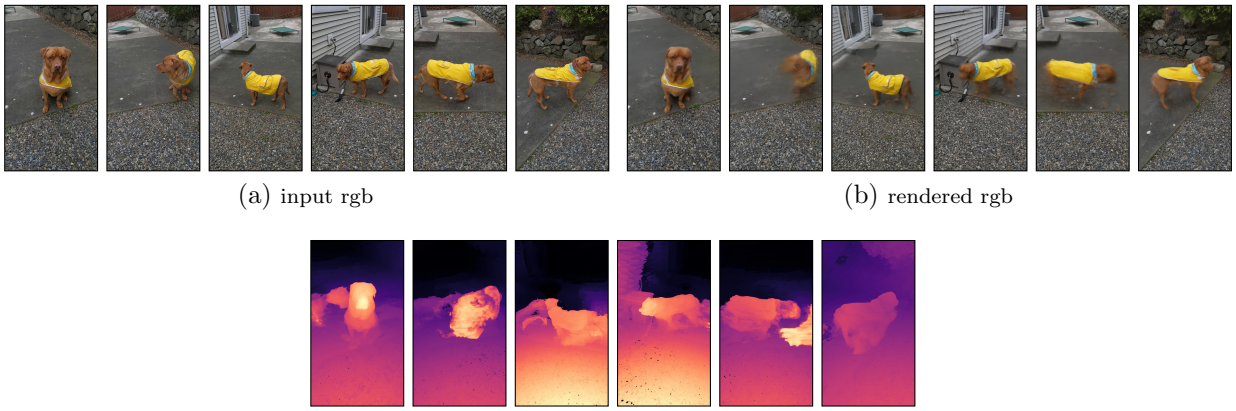


Figure 3.20: Topological changes break our method. The recovered geometry can be incorrect despite looking good from some viewpoints.

Rapid Motion: NeRF relies on seeing multiple observations to constrain where density lies in the volumes. In the presence of rapid motion, such as in Figure 3.21, certain states of the scene may only be visible for a short period of time making it harder to reconstruct.



(a) input rgb (b) rendered rgb

(c) rendered depth

Figure 3.21: This example shows Toby the dog moving around freely, showcasing two limitations of our method. (1) *Rapid motion*: Because Toby moves quite fast, the camera only sees him in certain poses for a short amount of time, resulting in a sparse set of observations for certain poses. This can make those poses under-constrained. (2) *Orientation flips*: Toby wanders back and forth, showing different sides of his body. Depending on which orientation Toby is modeled as in the template, it is difficult for the deformation field to predict a flipped orientation.

Orientation flips: Optimizations solving for any parameterization of rotations are known to be non-convex due to both Gauge ambiguity and the inherent "twistedness" of the space of SO(3) [226]. As a simple example to illustrate this, imagine trying to align two coins in 3D. If the coin is initialized in a flipped orientation where heads faces the tail side of the other coin, then the 'fit' of the two coins must get worse before getting better when rotating towards the global minimum.

We encounter the same issue when optimizing for our deformation fields. If the template of a scene is in a certain orientation, but the deformation field for an observation is initialized in the wrong orientation the method will get stuck in a local minima and result in sub-optimal alignment. We show an example of this in Figure 3.21 where frames with Toby's left side visible are reconstructed better than when Toby's right side is visible.

Hollow Face Illusion: The hollow-face illusion is an optical illusion where a concave (pushed in) imprint of an object appears to be convex (pushed out) instead.

A feature of this illusion is that the convex illusion appears to follow the viewer's eye.

Figure 3.22: If the user's gaze consistently follows the camera, the reconstructed *nerfie* represents the user's gaze as geometry, akin to the Hollow-Face illusion [71]. This is apparent in the depth map and makes the reconstructed model appear as if they are looking at the camera even when the geometry is fixed.

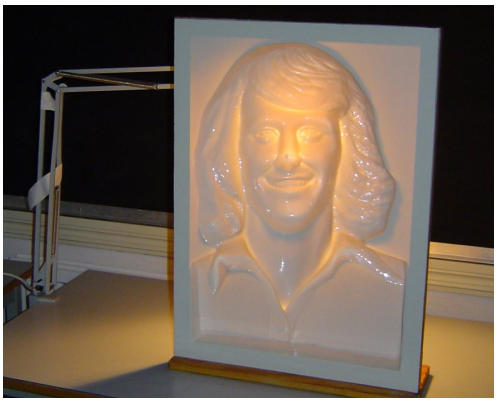


Figure 3.23: This concave mask of the Swedish tennis player Bjorn Borg appears to be convex due to the hollow-face illusion. By Skagedal, 2004 (Public Domain).

This illusion has been purposefully used in the Disneyland haunted mansion to create face busts which appear to follow you and in the popular T-Rex illusion [25]. We show an example of this illusion in Figure 3.23.

We observe that the ambiguity which causes this illusion can also be a failure more for our method. In Figure 3.22, we show an example where a user fixes their gaze in the direction of the camera while capturing themselves. Instead of modeling the eye motion as a deformation, our method models the eyes concavities as can be seen in the geometry.

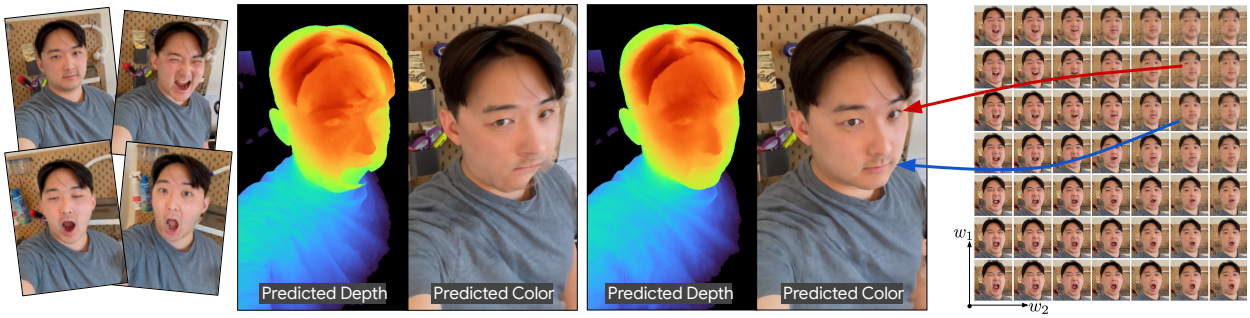
3.5 Conclusion

Deformable Neural Radiance Fields extend NeRF by modeling non-rigidly deforming scenes. We show that our as-rigid-as-possible deformation prior, and coarse-to-fine deformation regularization are the key to obtaining high-quality results. We showcase the application of casual selfie captures (*nerfies*), and enable high-fidelity reconstructions of human subjects using a cellphone capture. Future work may tackle larger/faster motion, topological variations, and enhance the speed of training/inference.

In the next chapter we will show how to model scenes with topological variations.

Chapter 4

CASUAL CAPTURE OF TOPOLOGICALLY VARYING SCENES



(a) Input Images (b) Nerfies $(\mathbf{c}, \sigma) = F(x, y, z)$ (c) HyperNeRF $(\mathbf{c}, \sigma) = F(x, y, z, w_1, w_2)$ (d) Hyper-Space

Figure 4.1: Nerfies can capture deformable scenes, but often struggles in the presence of significant deformation or topological variation, as evidenced in (a). By modeling a family of shapes in a high dimensional space, we can handle topological variation and produce more realistic renderings and more accurate geometric reconstructions, as can be seen in (b).

Many real-world motions involve changes in topology. Examples include cutting a lemon, or tearing a piece of paper. While not strictly a genus change, motions like closing your mouth cause changes in surface connectivity that can also be considered "topological". Such changes in topology often cause problems for algorithms that seek to reconstruct moving three dimensional scenes, as they cause motion discontinuities or singularities.

A clever approach for addressing topology changes is to represent the 3D scene as a *level set* in a 4D volume. Pioneered in the late 1980s [153], level set methods model moving scenes as static objects in a higher dimensional ambient space, and topological changes as *smooth* (rather than discontinuous) transformations.

In this paper, we adapt the level set framework for deformable neural radiance fields

[159], to generate photorealistic, free-viewpoint renderings of objects undergoing changes in topology. In doing so, we use modern tools like MLPs to significantly generalize the classical level set framework in key ways. First, whereas classical level sets add a single ambient dimension, we can add any number of ambient dimensions to provide more degrees of freedom. Second, rather than restrict level sets to hyper-planes, as is traditional, we allow general, curved slicing manifolds, represented through MLPs.

Our approach models each observation frame as a nonplanar slice through a hyperdimensional NeRF — a HyperNeRF. Previous methods using higher dimensional inputs require either substantial regularization or additional supervision. In contrast, our method retains a deformation field, which has previously demonstrated strong ability to fuse information across observations, and instead of regularizers, we use an optimization strategy that encourages smooth behavior in the higher dimensions. This enables our method to reconstruct high-quality geometry even when some poses are observed from only a small range of angles.

Our method enables users to capture photorealistic free-viewpoint reconstructions of a wide range of challenging deforming scenes from *monocular video*, e.g., waving a mobile phone in front of a moving scene. We demonstrate the quality of our method on two tasks: (i) interpolating smoothly between "moments" while maintaining visual plausibility, and (ii) novel-view synthesis with fixed moments. Our method, *HyperNeRF*, outperforms existing methods on both tasks by significant margins.

4.1 Related Work

4.1.1 Non-Rigid Reconstruction

A common approach in non-rigid reconstruction techniques is to decompose a scene into a canonical model of scene geometry (which is fixed across frames) and a deformation model that warps the canonical scene geometry to reproduce each input image. The difficulty of this task depends heavily on the inherent ambiguity of the problem formulation. Using only a monocular video stream is convenient and inexpensive, but also introduces significant

ambiguity which must be ameliorated through the use of factorization [24] or regularization [207]. On the opposite end of the spectrum, complicated and expensive capture setups using multiple cameras and depth sensors can be used to overconstrain the problem, thereby allowing 3D scans to be registered and fused to produce high quality results [43, 53]. Machine learning techniques have been used effectively for non-rigid reconstruction, when applied to direct depth sensors [21, 177]. Our method requires only monocular RGB images from a conventional smartphone camera as input — no depth sensors or multi-view capture systems are required.

Several works have used learning techniques to solve for deformation based models of shape geometry [96, 152], though because these works model only geometry and not radiance, they cannot be applied to RGB image inputs and do not directly enable view synthesis. Yoon *et al*. [237] use a combination of multi-view cues as well as learned semantic priors in the form of monocular depth estimation to recover dynamic scenes from moving camera trajectories. In contrast, our approach requires no training data other than the input sequence being used as input, as is typical in NeRF-like models. Neural Volumes [127] represents deformable scenes using a volumetric 3D voxel grid and a warp field, which are directly predicted by a convolutional neural network. As we will demonstrate, Neural Volumes's high fidelity output relies on the use of dozens of synchronized cameras, and does not generalize well to monocular image sequences. The Deformable NeRF technique of Park et al. [159] uses NeRF-like learned distortion fields alongside the radiance fields of Mildenhall et al. [143] to recover "nerfies" of human subjects, and is capable of generating photorealistic synthesized views of a wide range of non-stationary subjects. We build directly upon this technique, and extend it to better support subjects that not only move and deform, but that also vary topologically.

4.1.2 Neural Rendering

The nascent field of "neural rendering" aims, broadly, to use neural networks to render images of things. This is an emerging area of study that is changing rapidly, but progress in

the field up through 2020 is well-documented in the survey report of Tewari et al. [204]. The dominant paradigm in this field has, until recently, been framing the task of synthesizing an image as a sort of "image to image translation" task, in which a neural network is trained to map some representation of a scene into an image of that scene [89]. This idea has been extended to incorporate tools and principles from the graphics literature, by incorporating reflectance or illumination models into the "translation" process [139, 198], or by using proxy geometries [5, 206] or conventional renderings [58, 102, 135, 141] as a harness with which neural rendering can then be guided. Though these techniques are capable of producing impressive results, they are often hampered by a lack of consistency across output renderings — when rendering is performed independently by a "black box" neural network, there is no guarantee that all renderings of scene will correspond to a single geometrically-consistent 3D world.

Research within neural rendering has recently begun to shift away from this "image to image translation" paradigm and towards a "neural scene representation" paradigm. Instead of "rendering" images using a black box neural network that directly predicts pixel intensities, scene-representation approaches use the weights of a neural network to directly model some aspect of the physical scene itself, such occupancy [140], distance [156], surface light field [236], or a latent representation of appearance [189, 191]. The most effective approach within this paradigm has been constructing a neural radiance field (NeRF) of a scene, and using a conventional multilayer perceptron (MLP) to parameterize volumetric density and color as a function of spatial scene coordinates [143]. NeRF's usage of classical volumetric rendering techniques has many benefits in addition to enabling photorealistic view synthesis: renderings from NeRF must correspond to a single coherent model of geometry, and the gradients of volumetric rendering are well-suited to gradient-based optimization. Note that, in NeRF, a neural network is not used to render an image — instead, an analytical physics-based volumetric rendering engine is used to render a scene whose geometry and radiance happen to be parameterized by a neural network.

Though NeRF produces compelling results on scenes in which all content is static, it

fails catastrophically in the presence of moving objects. As such, a great deal of recent work has attempted to extend NeRF to support dynamic scenes. We will separate these NeRF variants into two distinct categories: *deformation-based* approaches apply a spatially-varying deformation to some canonical radiance field [159, 168, 209], and *modulation-based* approaches directly condition the radiance field of the scene on some property of the input image and modify it accordingly [60, 114, 119, 229].

Deformation-based NeRF variants follow the tradition established by the significant body of research on non-rigid reconstruction [149], and map observations of the subject onto a template of that subject. Nerfies [159], D-NeRF [168], and NR-NeRF [209] all define a continuous deformation field which maps observation coordinates to canonical coordinates which are used to query a template NeRF. Because multiple observations of the subject are used to reconstruct a single canonical template, and only the deformation field is able to vary across images, this approach yields a well-constrained optimization problem similar to basic NeRF. Similar to how NeRF parameterizes radiance and density using a coordinate-based MLP, these deformation-based NeRF variants use an MLP to parameterize the deformation field of the scene, and are thereby able to recover detailed and complicated deformation fields. However, this use of a *continuous* deformation field means that these techniques are unable to model any topological variations (e.g., mouth openings) or transient effects (e.g, fire). Topological openings or closings require a discontinuity in the deformation field at the seam of the closing, which MLPs cannot model easily. This is a consequence of the fact that coordinate-based MLPs with positional encoding perform interpolation with a band-limited kernel, when viewed through the lens of neural tangent kernels [202].

Modulation-based or *latent-conditioned* NeRF variants adopt the well established technique of conditioning a neural network with a latent code to modulate its output. 2D generative models such as GANs [69] or VAEs [105] input latent code to a 2D CNN which then outputs a corresponding image. This technique is also used to encode a space of 3D shapes; for example, DeepSDF [156] and IM-NET [39] modulate a signed distance field (SDF) based on input latent codes, while Occupancy Networks [140] modulate an occupancy field.

Similarly, SRNs [191] modulate a latent representation which is decoded by an LSTM.

Following these footsteps, instead of modeling the scene as a single NeRF that is warped to explain individual images, modulation-based NeRF techniques provide additional information (e.g., the image's timestamp or a latent code) as input to the MLP, directly changing the radiance field of the scene. As such, these techniques are capable of modeling any deformation, topological change, or even complex phenomena such as fire. However, because these modulated NeRFs may have completely different radiance and density across input images, these techniques result in a severely under-constrained problem and allows for trivial, non-plausible solutions. This issue can be addressed by providing additional supervision such as depth and optical flow (as in Video-NeRF [229] and NSFF [119]) or by using a multi-view input captured from 7 synchronized cameras (as in DyNeRF [114]). For facial avatars, Gafni et al. [60] avoids this issue by using a face-centric coordinate frame from a 3D morphable face model [205]. NeRF in the Wild also uses a similar modulated approach (albeit for a different task) in which latent codes are optimized and provided as input to an MLP, which also introduces ambiguities that are addressed by allowing those codes to only modify radiance but not density [136].

Our method can be thought of as a combination of deformation-based and modulation-based approaches: we use deformations to model motion in the scene, resulting in a well-behaved optimization, but we also extend NeRF's 3D input coordinate space to take additional higher-dimension coordinates as input, and allow for deformations along the higher dimensions as well as the spatial dimensions. As we will show, this approach is able to use the higher dimensions to capture changes in object topology, which a strictly deformation-based approach would be unable to model.

4.2 Modeling Time-Varying Shapes

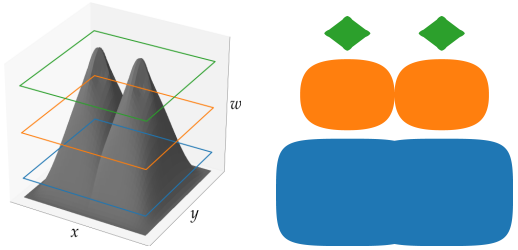
Our method represents changes in scene topology by providing a NeRF with a higher-dimensional input. To provide an intuition and a justification for our formulation in higher dimensions, this section introduces level set methods which our method draws inspiration

from. Note that our method is *not* a level set method, though we will apply intuitions gained from these examples to our NeRF setting.

There are two common approaches for mathematically representing the surface of an object: a surface can be defined *explicitly*, perhaps with a polygonal mesh, or *implicitly*[1], perhaps as the level set of a continuous function. Explicit representations of shape, though effective and ubiquitous, are often poorly suited to topological variation of a surface: slicing a polygonal mesh into two halves, for example, likely requires creating new vertices and redefining the edge topology of the mesh. This sort of variation is particularly difficult to express in the context of gradient-based optimization methods such as NeRF, as this transformation is discontinuous and therefore not easily differentiated. In contrast, implicit surfaces provide a natural way to model the evolution of a surface in the presence of topological changes, such as when the surface develops a hole or splits into multiple pieces.

4.2.1 Level Set Methods

Level set methods model a surface implicitly as the zero-level set of an auxiliary function [153]. For example, a 2D surface can be defined as $\Gamma = \{(x, y) | S(x, y) = 0\}$, where $S : (x, y) \rightarrow s$ is a signed-distance function with $s > 0$ for points inside the surface, and $s < 0$ for points outside the surface. To model a surface that varies topologically with respect to some additional dimension (such as "time"), one can add an additional dimension w, and thereby define a 3D surface $\Gamma = \{(x, y, w) | S(x, y, w) = 0\}$. We will refer to the space by these additional dimensions as



(a) Aux. Function (b) 2D Sliced Shapes

Figure 4.2: Level-set methods provide a means to model a family of topologically-varying shapes (b) as slices of a higher dimensional auxiliary function (a).

[1]Confusingly, NeRF-like models, or coordinate-based approaches in general, are sometimes referred to in the literature as "implicit" models, in reference to the idea that they encode scenes "implicitly" using the weights of a neural network. This new use of "implicit" is distinct from its common use in the literature, so to avoid confusion we will only use "implicit" according to its meaning in the level set literature.

the "ambient" space. The 2D surface at some w_i ambient coordinate can then be expressed as the 2D cross-section of the 3D surface Γ obtained by slicing it with the plane passing through $w = w_i$. See Figure 4.2 for an illustration.

This idea can be extended to learn a collection of shapes. Given a set of implicitly-defined 2D shapes, one can learn an SDF Γ which contains all such 2D shapes as individual slices of a canonical 3D surface. Following DeepSDF [156], let us define $S : (x, y, w) \rightarrow s$ as an MLP, where the per-shape ambient coordinates $\{w_i\}$ are learned as an embedding layer [17]. The weights of the MLP and the hyper coordinates can then be optimized using gradient descent. Figure 4.6a shows an example of four different shapes being encoded in a single 3D SDF in this way. As with time-varying shapes, this approach gives us a natural way to interpolate between the shapes by interpolating between the learned slicing planes. As shown by DeepSDF, this formulation can be extended to an arbitrary number of spatial and ambient dimensions. For example, by formulating this same problem with 3 spatial dimensions and 256 ambient dimensions, we can learn 3D shapes as 3-dimensional cross sections of a 259-dimensional hyper-surface.

4.2.2 Deformable Slicing Surfaces

Level set methods work by "slicing" through a function, usually with an axis-aligned plane. The plane of this slice spans the spatial axes (the x and y axes in our 2D/3D example), and occupies a single ambient coordinate (the w axis in our example) at all points. A consequence of using an axis-aligned slice is that every desired output shape must exist as a cross section cut by the slicing plane. In certain circumstances, this can lead to an inefficient use of space. For

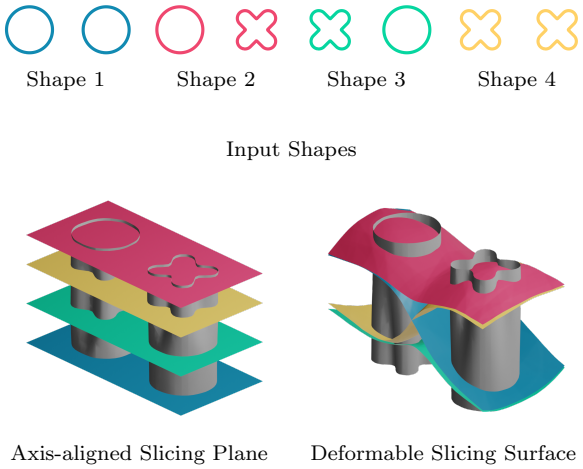


Figure 4.3: Level-set methods model shapes as slices of a higher-dimensional surface. *Axis-aligned planes* (AP) slice the surface perpendicular to the higher-dimensional axis; *deformable surfaces* (DS) can reference varying parts of hyper-space.

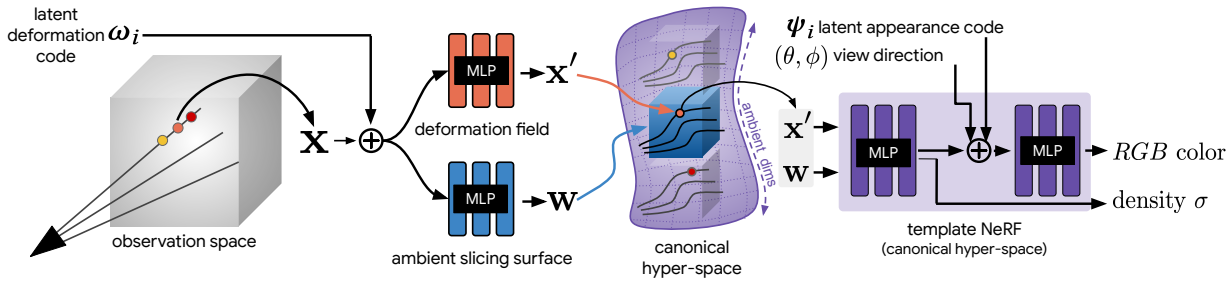


Figure 4.4: An overview of our model architecture. We associate a latent deformation code $\boldsymbol{\omega}_i$ and a latent appearance code $\boldsymbol{\psi}_i$ with each image i. Rays are cast from the camera in the space, and samples $\mathbf{x}$ along those rays are then concatenated with the image's latent deformation code $\boldsymbol{\omega}_i$ and provided as input to MLPs parameterizing a deformation field, which yields a warped coordinate $\mathbf{x}'$ and a slicing surface in hyperspace, which yields a coordinate in our ambient space $\mathbf{w}$. both outputs are concatenated to yield a mapped point $(\mathbf{x}', \mathbf{w}) = (x, y, z, w_1, w_2, \ldots)$. The concatenation of $(\mathbf{x}', \mathbf{w})$, the viewing direction $\mathbf{d} = (\theta, \phi)$, and the appearance code $\boldsymbol{\psi}_i$ are then used as inputs to the MLP parameterizing the template NeRF. The densities and colors produced by that MLP are then integrated along the ray as per the physics of volumetric rendering, as in Mildenhall et al. [143].

example, consider the set of shapes shown in Figure 4.3a, which are different permutations of the same two shapes. If axis-aligned slices are used, the ambient surface must contain a *copy* of each of the four permutations as shown in Figure 4.6a. This is inefficient considering there are only two possible sub-shapes–a circle and a cross.

To address this, we allow the slicing surface to deform, such that the output shape is the cross-section sliced by a non-planar *deformable* slicing surface. This allows different spatial locations to reference different parts of the ambient coordinate space, resulting in a more compact representation in hyperspace. We define the deformable slicing surface as $H : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{w}$, where $\mathbf{x}$ is a spatial position, $\mathbf{w}$ is a position along the ambient axes, and $\boldsymbol{\omega}_i$ is a per-input latent embedding (whose dimensionality and meaning need not match that of the ambient coordinate). The SDF is then queried at the coordinate obtained by concatenating $\mathbf{x}$ and $H(\mathbf{x}, \boldsymbol{\omega}_i)$. As shown in Figure 4.6b, this parameterization is able to slice through the ambient dimensions to model arbitrary mixtures of shapes.

4.3 Method

Here we describe our method for modeling non-rigidly deforming scenes given a casually captured monocular image sequence. The focus of our method is to be able to accurately model the appearance and geometry of *topologically varying* scenes. Before we introduce HyperNeRF, we first review its foundations: neural radiance fields (NeRF) [143] as well as unconstrained and deformable extensions of it [136, 159], all of which we will build upon.

4.3.1 Review of NeRF, NeRF-W, and Deformable NeRF

NeRF represents a scene as a continuous, volumetric field of density and radiance, defined as $F : (\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma)$. The function F is parameterized by a multilayer perception (MLP) and maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\phi, \theta)$ to a color $\mathbf{c} = (r, g, b)$ and density σ. Instead of directly providing input coordinates to the MLP directly, NeRF first maps each input $\mathbf{x}$ (and $\mathbf{d}$) using a sinusoidal positional encoding:

$$\gamma(\mathbf{x}) = \left[\sin(\mathbf{x}), \cos(\mathbf{x}), \sin(2\mathbf{x}), \cos(2\mathbf{x}), \ldots, \sin(2^{m-1}\mathbf{x}), \cos(2^{m-1}\mathbf{x})\right]^{\mathrm{T}}, \quad (4.1)$$

where m is a hyper-parameter that controls the number of sinusoids used by the encoding. As shown in Tancik et al. [202], this encoding allows the MLP to model high-frequency signals in low frequency domains, where the parameter m serves to control smoothness of the learned representation by modifying the effective bandwidth of an interpolating kernel.

Appearance Variation To handle unconstrained "in the wild" images, the MLP in NeRF can be additionally conditioned on an appearance embedding $\boldsymbol{\psi}_i$ for each observed frame $i \in \{1, \ldots, n\}$, as shown in Martin-Brualla et al. [136]. This allows NeRF to handle appearance variations between input frames, such as those caused by illumination variation or changes in exposure and white balance.

Deformations Because a NeRF is only able to represent a static scene, we use the SE(3) formulation of a deformation field proposed in Nerfies [159] to model non-rigid motion.

Nerfies defines a mapping $T : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{x}'$ that maps all observation-space coordinates $\mathbf{x}$ to canonical-space coordinates $\mathbf{x}'$, conditioned on a per-observation latent deformation code $\boldsymbol{\omega}_i$. The deformation field T is parameterized by an MLP $W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow (\mathbf{r}, \mathbf{v})$, where $(\mathbf{r}, \mathbf{v}) \in \mathfrak{se}(3)$ encode the rotation and translation. To encourage deformations to be as-rigid-as-possible, Nerfies proposes an elastic regularization loss which penalizes non-unit singular values of the Jacobian of the deformation field. We did not find the elastic loss well suited when modeling topological variations, as they directly violate the rigidity prior. Please see Park et al. [159] for details.

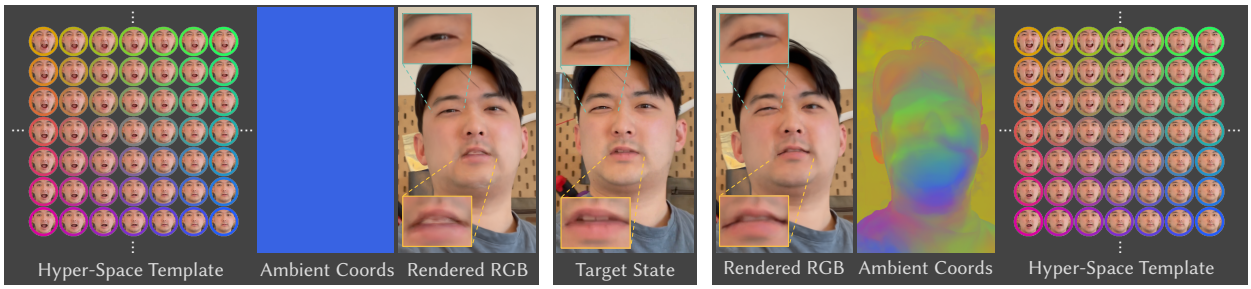
Windowed Positional Encoding Tancik et al. [202] showed that the number of frequencies m in the positional encoding γ controls the bandwidth of an MLP's Neural Tangent Kernel (NTK) [92]. A small value for m results in a smooth estimator that may under-fit the data, while a large value of m may result in over-fitting. Park et al. [159] and Hertz et al. [79] use this property to implement a coarse-to-fine strategy when optimizing an MLP, by slowly narrowing the bandwidth of the NTK by weighting the frequency bands of the positional encoding with a window function such as the one used by Park et al. [159]:

$$w_j(\alpha) = \frac{1 - \cos(\pi \operatorname{clamp}(\alpha - j, 0, 1))}{2}, \tag{4.2}$$

where $j \in \{0, \ldots, m-1\}$ is the index of the frequency band of the positional encoding. Linearly increasing $\alpha \in [0, m]$ is equivalent to sliding a truncated Hann window down the frequency bands of positional encoded features. The windowed positional encoding is then computed as:

$$\gamma_\alpha(\mathbf{x}) = \left[w_0(\alpha)\sin(\mathbf{x}), \ldots, w_{m-1}(\alpha)\cos(2^{m-1}\mathbf{x})\right]^{\mathrm{T}}. \tag{4.3}$$

Nerfies uses this to optimize the deformation field in a coarse-to-fine manner which prevents getting stuck in sub-optimal local minima while being retaining the ability to represent high-frequency deformations [159]. We do the same for our spatial deformation field.



(a) Axis-Aligned Slicing Plane (AP) (b) (c) Deformable Slicing Surface (DS)

Figure 4.5: We render the target state (b) from a novel view using axis-aligned slicing planes (AP, a) and deformable surfaces (DS, c). We show the hyper-space template rendered at ambient coordinates, sampled on a regular grid. We plot the ambient coordinate of each pixel — the color of the coordinates correspond to the outlined color of each template sample. AP must wholly model each scene state and results in artifacts in the eyes, mouth, and chin. DS can more efficiently use the template since each part of the scene can refer to different parts of the template, resulting in sharper details.

4.3.2 Hyper-Space Neural Radiance Fields

Motion in a scene can be divided into two categories: (a) motions that preserve the topology of the scene; and (b) motions which change the apparent topology of the scene. Here we use the term "topology" in the sense we defined in the beginning of the chapter.

Deformation fields can effectively model topology-preserving motion with one smooth deformation field. On the other hand, a continuous deformation field cannot easily model changes in topology. This is because a change in topology necessarily involves a discontinuity in the deformation field. Our deformation fields are continuous, by virtue of being encoded within the weights of an MLP that behaves as a smooth interpolator, and therefore cannot represent such discontinuities. See Figure 4.6 for a visual explanation.

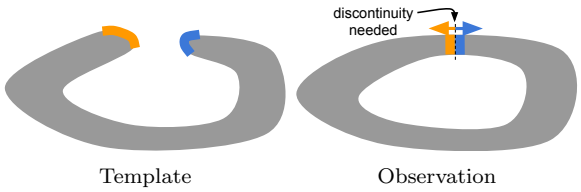


Figure 4.6: An example topological change where the ring opens at the marked seam. A deformation referencing the template for each point in the observation frame requires a discontinuity at the seam, where an infinitesimal step from the orange position towards the blue position results in a big change. If the template were the closed ring, the contents of the ring would be inaccessible using a deformation.

Here we present our method, *HyperNeRF*, which extends neural radiance fields into higher

dimension to allow topological variations. In §4.2.1 we introduced the level set method, which can naturally model topologically changing shapes as cross-sections of a higher-dimensional ambient surface. We use the same idea in the context of neural radiance fields. Our architecture is visualized in Figure 4.4.

Hyper-space Template Deformable NeRFs [159] represent the scene in a canonical-space *template* NeRF which is indexed by a spatial deformation field to render observation frames. Our idea is to embed the template NeRF in higher dimensions, where a slice taken by an intersecting high-dimensional slicing surface yields a full 3D NeRF. Note that we are still rendering 3-dimensional scenes, and thus ray casting and volume rendering occur in the same manner as a normal NeRF.

We extend the domain of the template NeRF to a higher dimensional space: $(\mathbf{x}, \mathbf{w}) \in \mathbb{R}^{3+W}$, where W is the number of higher dimensions. The formulation of the template NeRF is otherwise identical, with it being represented as an MLP

$$F : (\mathbf{x}, \mathbf{w}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma)\,. \tag{4.4}$$

Slicing Surfaces In §4.2.2 and Figure 4.3 we showed how an ambient surface could be cut with a slicing surface to obtain an output shape. In the same way, we can take cross-sections of a hyper-space NeRF. The 2D example only had a single ambient dimension, and thus the cross-section was obtained using a single slicing surface intersecting the higher dimensional axis at a point. To obtain a cross-section in the presence of more than one higher dimension, we instead have a slicing surface for each higher dimension.

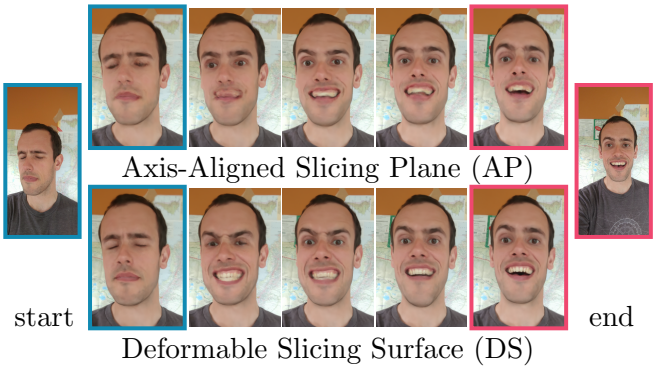
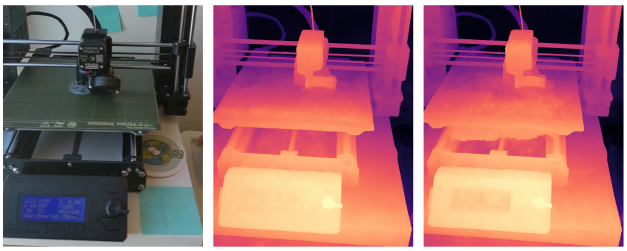


Figure 4.7: Novel views synthesized by linearly interpolating the embedding $\boldsymbol{\omega}_i$ of two frames from "Expressions 1". With (AP), all spatial points are rendered at the same ambient coordinate resultin in blurry-cross-fading artifacts during interpolation. With (DS), different spatial positions to reference varying parts of the ambient space, resulting in better interpolations.

This is the geometric interpretation of simply fixing the values of $\mathbf{w}$, which leaves the span of the spatial axes as the cross-section along the ambient dimension.

The most basic implementation of this idea would be to learn a per-observation slicing planes, where we learn a set of coordinates $\{\mathbf{w}_i\}$ for each input observation as an embedding layer. This representation fixes all ray samples for an observation i to live in the same 3D subspace, analogous to the axis-aligned slicing plane introduced in §4.2.2. We found that this led to an inefficient use of the representation capacity of the template NeRF since topological variations happening in spatially distance parts of the scene must be copied across multiple sub-spaces to represent different combinations of poses. This can decrease reconstruction quality, and causes cross-fading artifacts when interpolating between moments. We therefore use deformable slicing surfaces, introduced in §4.2.2, which allow the slicing surface to deform, allowing different spatial positions to be mapped to different coordinates along the ambient dimensions. This results in a more efficient use of the ambient dimensions, and smoother interpolations between moments. See Figure 4.7 for a comparison between both methods, and Figure 4.6b for a 2D toy example illustrating the same concept in the context of level sets.

Similarly to the spatial deformation field, we define a deformable slicing surface field using an MLP. Each observation-space sample point $\mathbf{x}$ is mapped through the mapping $H : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{w}$, where $\boldsymbol{\omega}_i$ is the latent deformation code (shared with the spatial deformation field), and $\mathbf{w}$ is a point in the ambient coordinate space which defines the cross-sectional subspace for the sample. Given the template NeRF F, spatial deformation field T, and the slicing surface field H, the observation-space radiance field can be



(a) Input Color (b) Ours (c) [159]

Figure 4.8: Although not technically a change in topology, the bed of the 3D printer moving relative to its base requires a sharp change in the deformation field, resulting in artifacts. *HyperNeRF* alleviates these artifacts.

evaluated as:

$$\mathbf{x}' = T(\mathbf{x}, \boldsymbol{\omega}_i)\,, \tag{4.5}$$

$$\mathbf{w} = H(\mathbf{x}, \boldsymbol{\omega}_i)\,, \tag{4.6}$$

$$(\mathbf{c}, \sigma) = F(\mathbf{x}', \mathbf{w}, \mathbf{d}, \boldsymbol{\psi}_i)\,. \tag{4.7}$$

where in practice, we apply separate positional encodings to each input $\mathbf{x}$, $\mathbf{d}$, and $\mathbf{w}$. We use the windowed positional encoding γ_α for the deformation field T and slicing surface field H.

We do not use the identity concatenation of the positional encoding for the ambient coordinates. Barron et al. [9] showed that the identity encoding does not meaningfully affect performance or speed, and omitting it confers a specific advantage: we can collapse ambient dimensions during the initial phases of the optimization by setting $\alpha = 0$, as described in the previous section.

Delayed use of Ambient Dimensions Motion can be encoded by deforming points using the spatial deformation field T, or by moving along the ambient dimensions using the slicing surface H. If a motion does not involve a change in topology, we prefer using a spatial deformation over moving through the ambient dimensions. This is because spatial deformations result in a more constrained optimization, since several observations get mapped to the same point on the template. In addition, spatial deformations result in better interpolations and they only involve movement and cannot directly change the density of the template. We therefore delay the use of the ambient dimensions by using the windowed positional encoding (§4.3.1) on the ambient coordinates $\mathbf{w}$. We disable the identity concatenation for the positional encoding so that the input can be completely turned off when $\alpha = 0$. We fix $\alpha = 0$ for a number of iterations and then linearly ease in the rest of the frequencies. This windowing is only applied to the ambient dimensions. The spatial coordinate $\mathbf{x}$ is not windowed, and the spatial deformation field uses its own windowing schedule from Park et al. [159].

Figure 4.9: Here we show qualitative comparisons of Nerfies [159] (top rows) with HyperNeRF (ours) (bottom rows) on four different image sequences. For each sequence we visualize two different moments, each with a different apparent topology. Nerfies is unable to model the topological variation with its deformation field and therefore distorts the geometry in implausible ways in order to explain the training data, while HyperNeRF produces more plausible geometry estimates and more accurate renderings on novel views not seen during training.

4.4 Experiments

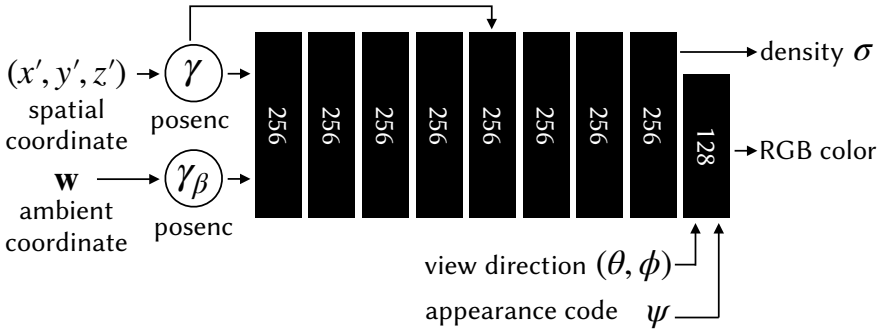


Figure 4.10: A diagram of our hyper-space template network, which is identical to the original NeRF MLP, except it takes an additional ambient coordinate and an appearance latent code $\boldsymbol{\psi}$ as in Martin-Brualla et al. [136]

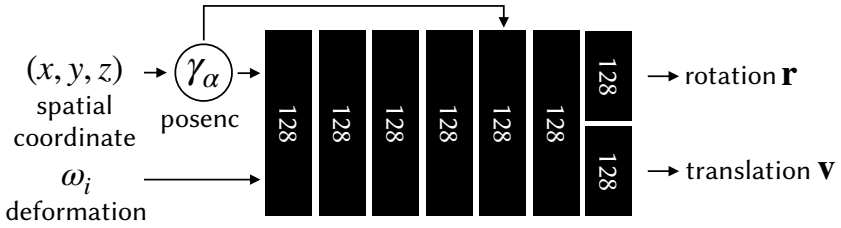


Figure 4.11: A diagram of our deformation network. It is identical to the deformation MLP of Nerfies [**?**].

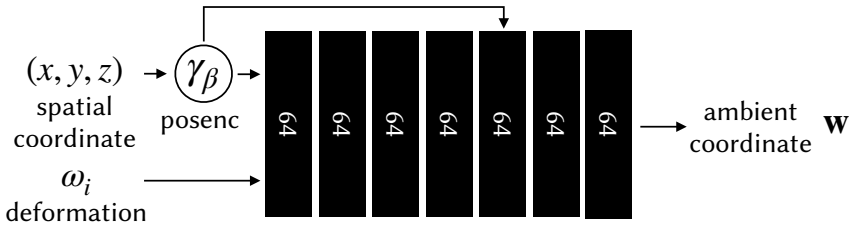


Figure 4.12: A diagram of our ambient slicing surface network. It shares the input deformation code $\boldsymbol{\omega}_i$ with the deformation network. The ambient slicing surface network uses a different windowed positional encoding parameter β, and directly outputs an ambient coordinate $\mathbf{w}$.

4.4.1 Implementation Details

Our implementation of NeRF and the deformation field closely follows Nerfies [159]. As in NeRF [143], we only use an L2 photometric loss. We use 8 dimensions for the latent appearance and deformations codes. We exponentially decay our learning rate from 10^{-3} to 10^{-4}. For the windowed positional encoding parameter α of the deformation field, we follow the same easing schedule as in Park et al. [159]. We implement the deformable slicing surface as an MLP with depth 6 and width 64 with a skip connection at the 5th layer. The last layer is initialized with weights sampled from $\mathcal{N}(0, 10^{-5})$. We use $m = 1$ for the hyper-coordinate input $\mathbf{w}$ to the template T, fixing $\beta = 0$ for 1000 iterations and then linearly increase it to 1 over 10k iterations. We use $m = 6$ for the spatial position input $\mathbf{x}$ to the slicing surface field H. We use 2 ambient dimensions ($W = 2$) for all experiments, as increasing W did not improve performance for our sequences.

We implement our method on top of JaxNeRF [48], a JAX [22] implementation of NeRF. For our evaluation metrics, we train at half of 1080p resolution (960x540) for 250k iterations, using 128 samples per ray with a batch size of 6,144, which takes roughly 8 hours on 4 TPU v4s. For qualitative results, we train at full-HD (roughly 1920x1080) with 256 samples per ray for 1M iterations, which takes roughly 64 hours.

4.4.2 Evaluation

Here we analyze the performance of our method both quantitatively and qualitatively. To best judge quality, we urge the reader to view the supplementary video which contains many visual results.

Quantitative Evaluation

We evaluate our method on two tasks: (i) how well it interpolates between different moments seen during training while maintaining visual plausibility; and (ii) its ability to perform novel-view synthesis. We collected our own sequences for these tasks, as existing datasets

do not focus on topologically varying scenes. While the dataset of Yoon et al. [237] contains two sequences exhibiting topological changes, these sequences have short capture baselines (∼1.1m) and exaggerated frame-to-frame motion due to aggressive temporal sub-sampling.

We measure visual quality with LPIPS [242], MS-SSIM [222], and PSNR. Note that because we are reconstructing dynamic scenes with a single moving camera, there can be ambiguities which cause slight differences from the true geometry or appearance of the scene. Because of this, we find that quantitative metrics often do not reflect perceptual quality, as we show in Figure 4.13. In particular, PSNR is incredibly sensitive to small shifts, and will penalize sharp images over blurry results, while MS-SSIM may not pick up artifacts which are obvious to humans. Out of the three metrics, we find that LPIPS best reflects perceptual quality.

Ablations We also compare with a couple variants of HyperNeRF: "HyperNeRF (DS)" uses the deformable slicing surface (§4.2.2), "HyperNeRF (DS, w/ elastic)" uses the elastic regularization loss from Park et al. [159], "HyperNeRF (AP)" uses the axis-aligned slicing plane, and "HyperNeRF (w/o deform)" removes the deformation field and only uses the hyper-space formulation.

Table 4.1: Metrics on the interpolation task, averaged across all sequences. We report PSNR, MS-SSIM, LPIPS, and the "average" metric of Barron et al. [9]. We color each row as **best**, **second best**, and **third best**. See Sec. 4.4.2 for details, and the supplement for metrics for each sequence.

	PSNR↑	MS-SSIM↑	LPIPS↓	Avg↓
NeRF [143]	21.3	.745	.490	.126
NV [127]	25.3	.880	.214	.0630
NSFF [119]	25.5	.863	.242	.0663
Nerfies [159]	27.7	.908	.193	.0487
Nerfies (w/o elastic)	28.0	.909	.193	.0476
Hyper-NeRF (DS)	28.3	.914	.185	.0456
Hyper-NeRF (DS, w/ elastic)	28.1	.912	.195	.0471
Hyper-NeRF (AP)	28.3	.912	.208	.0476
Hyper-NeRF (w/o deform)	27.4	.895	.253	.0557

Interpolation Like Nerfies [159], our method can interpolate between moments in an input video by interpolating the input embeddings $\boldsymbol{\omega}_i$ and $\boldsymbol{\psi}_i$. Here, we evaluate quality by leaving out intermediate frames from the input video and comparing them with corresponding interpolated frames. Specifically, we collect a set of videos, each 30-60s long, sub-sampled to 15fps and register the frames

Table 4.2: Quantitative evaluation on validation rig captures. We compare with baselines and ablations of our method. We color each row as **best**, **second best**, and **third best**. Note that traditional metrics like PSNR and SSIM are sensitive to small shifts, penalizing sharp images over blurry results (see Figure 4.13 below). See §4.4.2 for more details on these experiments.

	Broom (197 images)			3D printer (207 images)			Chicken (164 images)			Expressions (259 images)			Peel Banana (513 images)			Mean		
	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓	PSNR↑	MS-SSIM↑	LPIPS↓
NeRF [143]	19.9	.653	.692	20.7	.780	.357	19.9	.777	.325	20.1	.697	.394	20.0	.769	.352	20.1	.735	.424
NV [127]	17.7	.623	.360	16.2	.665	.330	17.6	.615	.336	14.6	.672	.276	15.9	.380	.413	16.4	.591	.343
NSFF [119]†	26.1	.871	.284	27.7	.947	.125	26.9	.944	.106	26.7	.922	.157	24.6	.902	.198	26.4	.917	.174
Nerfies [159]	19.2	.567	.325	20.6	.830	.108	26.7	.943	.0777	21.8	.802	.150	22.4	.872	.147	22.1	.803	.162
Nerfies (w/o elastic)	19.4	.581	.323	20.2	.820	.115	26.0	.935	.0837	21.8	.800	.149	21.7	.852	.157	21.8	.798	.165
Hyper-NeRF (DS)	19.3	.591	.296	20.0	.821	.111	26.9	.948	.0787	21.6	.800	.148	23.3	.896	.133	22.2	.811	.153
Hyper-NeRF (DS, w/ elastic)	19.5	.605	.277	20.2	.823	.109	27.5	.954	.0756	21.9	.806	.144	22.7	.882	.133	22.3	.814	.148
Hyper-NeRF (AP)	19.6	.596	.319	20.0	.814	.131	27.2	.950	.0941	22.2	.817	.149	22.4	.874	.142	22.2	.810	.167
Hyper-NeRF (w/o deform)	20.6	.714	.613	21.4	.846	.212	27.6	.950	.108	22.0	.793	.196	24.3	.914	.170	23.2	.843	.260

using COLMAP [178].

We build our dataset by using every 4th frame as a training frame, and taking the middle frame between each pair of training frames as a validation frame. Since certain frames may have been dropped due failed registration in COLMAP, we use the timestamp of the validation frame to compute its relative position between its corresponding training frames.

We compare our method with Nerfies [159], Neural Volumes [127], and NSFF [119]. For Neural Volumes, we render the interpolated frame by encoding the two reference frames with the encoder and linearly interpolating the latent codes. For NSFF, we linearly interpolate the input time variable between the two reference frames. Tab. 4.1 shows that our method outperforms all three baselines in most cases. Please see Figure 4.14 and Figure 4.15 for qualitative examples from each interpolation sequence.

Novel-view Synthesis We evaluate the novel-view synthesis quality of our method on topologically varying scenes. We render the scene at fixed moments from unseen viewpoints and compare how well the predicted images match the corresponding ground truth image. To allow for a fair comparison with prior work, we closely follow the evaluation protocol of Park et al. [159]. We create a capture rig comprised of a pole with two Pixel 3 phones rigidly

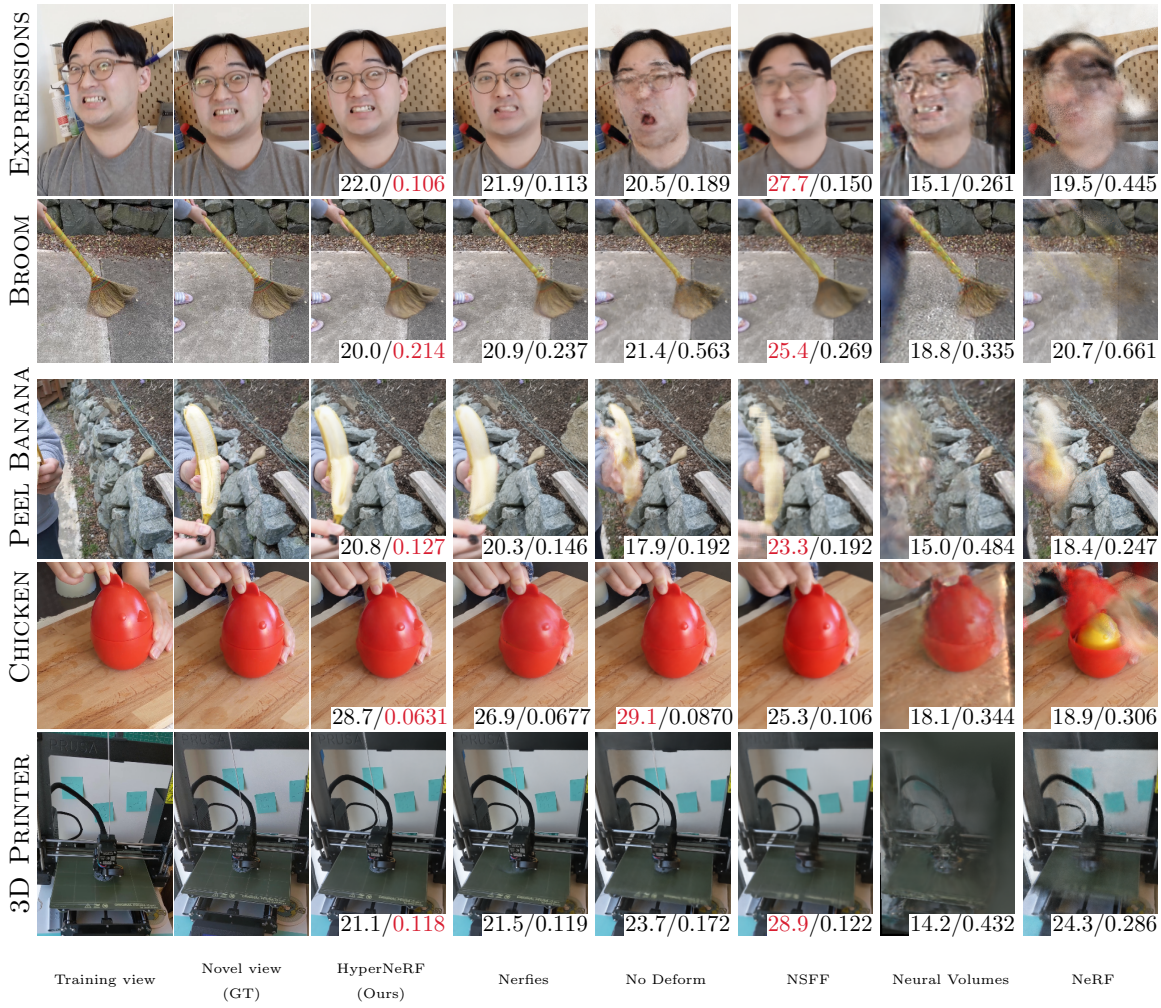


Figure 4.13: Comparisons of baselines and our method on validation rig scenes. PSNR / LPIPS metrics on bottom right with best colored red. Baselines shown are: Nerfies [159], HyperNeRF without deformations, Neural Volumes [127], NSFF [119], and NeRF [143]. Note how PSNR often prefers blurry results.

attached roughly 16cm apart. Please see Park et al. [159] for the details on dataset processing. We evaluate on the Broom sequence from [159] which exhibits a topological change when the broom contacts the ground, and augment the sequences of Park et al. [159] with 4 additional sequences exhibiting topological changes: 3D Printer, Chicken, Expressions, and Peel Banana. Tab. 4.2 reports the metrics computed on unseen validation views.

Baselines For view-synthesis, we compare with NeRF [143], Neural Volumes [127] and two recent dynamic NeRF methods: NSFF [119] and Nerfies [159]. For interpolation, we compare with NeRF, Nerfies, and Neural Volumes. To evaluate Neural Volumes on the interpolation task, we compute the latent codes of the two training images corresponding to each validation image with the image encoder, and linearly interpolate the encoding vector using the identical procedure mentioned in §4.4.2. We do not compare with Yoon et al. [237] as their code was not available. Instead, we use NSFF [119] which achieves higher quality on the dataset of Yoon et al. [237].

Qualitative Results

Figure 4.9 shows qualitative results, comparing our method with Park et al. [159]. We also show visual results from the validation rig dataset in Figure 4.13. Note how some of the image quality metrics sometimes prefer blurry results (e.g. Expressions).

Like our method, Nerfies [159] is able to produce sharp results with few artifacts. However, our method better reconstructs the poses of objects in scenes such as Peel Banana and Expressions in Figure 4.13. This is also shown by the warped faces in Figure 4.9, where Nerfies often does not model geometry for the chin.

4.4.3 2D Level Set Experiments

Here we describe our method for learning a family of topologically varying 2D signed distance functions (SDF). This method was used to generate Fig. 3.

We used truncated signed distance functions, truncated to range between -0.05 and 0.05. We learn a template MLP

$$F : (x, y, w) \rightarrow s\,, \tag{4.8}$$

which takes as input a normalized 2D coordinate $(x, y) \in [-1, 1]^2$ an ambient coordinate $w \in \mathbb{R}$, and outputs a signed distance s. This function defines a 3D surface which is sliced by a slicing surface. This be an axis-aligned plane, defined by a value of w defining a plane

that spans the x and y axes, or a deformable slicing surface defined by an ambient slicing surface MLP

$$H : (x, y, \omega_i) \to w \,, \tag{4.9}$$

where (x, y) are again the spatial coordinates, $\boldsymbol{\omega}_i$ is a per-shape latent code, and w is the output ambient coordinate. We show a detailed diagram of the 2D template in Figure 4.16 and deformable slicing surface in Figure 4.17.

Training Training batches are generated by randomly sampling points from the continuous, truncated SDFs of each shape i, resulting in data of the form $\left\{((x, y, \boldsymbol{\omega}_i), s)\right\}$. We use a Pseudo-Huber loss [33] function:

$$L_\delta(s - s^*) = \delta^2 \left(\sqrt{1 + \left(\frac{s - s^*}{\delta} \right)^2} - 1 \right) \,, \tag{4.10}$$

where s is the predicted SDF value, s^* is the ground truth SDF value, and δ is a hyperparameter that controls the steepness which we set to $\delta = 0.005$.

Implementation Details We use a positional encoding with a minimum degree as well as a maximum degree:

$$\gamma(\mathbf{x}) = \left[\sin(2^{m_-}\mathbf{x}), \cos(2^{m_-}\mathbf{x}), \ldots, \sin(2^{m_+-1}\mathbf{x}), \cos(2^{m_+-1}\mathbf{x}) \right]^{\mathrm{T}} \,, \tag{4.11}$$

where m_- is the minimum degree and m_+ is the maximum degree of the positional encoding. For the template MLP we $m_- = -2$ and $m_+ = 3$, for the ambient slicing surface MLP we use $m_- = -2$ and $m_+ = 2$. We train using the Adam optimizer with a fixed learning rate of 10^{-3} for 2000 iterations with a batch size of 512 which takes around a minute on a Google Colab TPU.

Interpolating Shapes We generate interpolated shaped by linearly interpolating the shape code ω_i between instances.

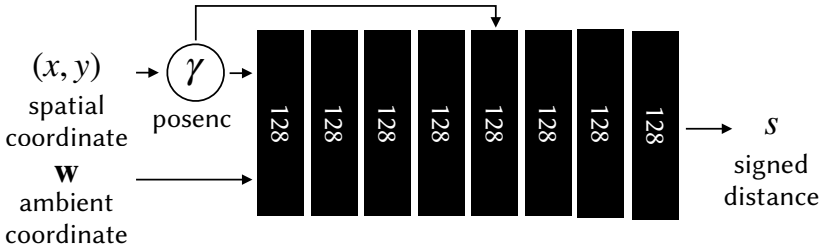


Figure 4.16: A diagram of architecture for the 2D template MLP used in the 2D level set experiments.

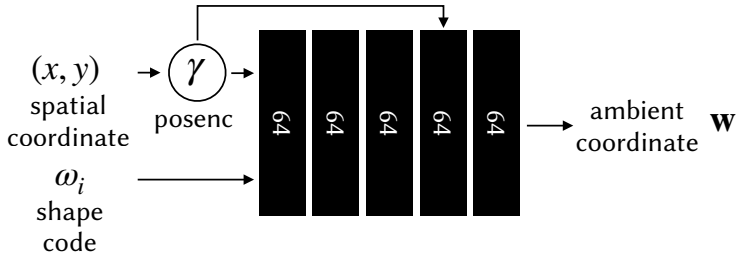


Figure 4.17: A diagram of architecture for the ambient slicing surface MLP used in the 2D level set experiments.

4.5 Limitations

As with all NeRF-like methods, camera registration affects the quality of the reconstruction. And, since our model uses only color images as input, imposing no domain-specific priors, we can only reconstruct what is observed. So, moments which are not captured well in the training data – such as when there is rapid motion – cannot be reconstructed by our method. We consider these limitations as fruitful avenues for future work.

Figure 4.14: Qualitative comparisons of our method, ablations, and baselines on the interpolation dataset (See §4.4.2). The interpolated frame is rendered by linearly interpolating between the latent codes of the two reference frames. The bottom row of each sequence shows the absolute error. Baselines shown are: Nerfies [159], Neural Volumes [127], NSFF [119], and NeRF [143].

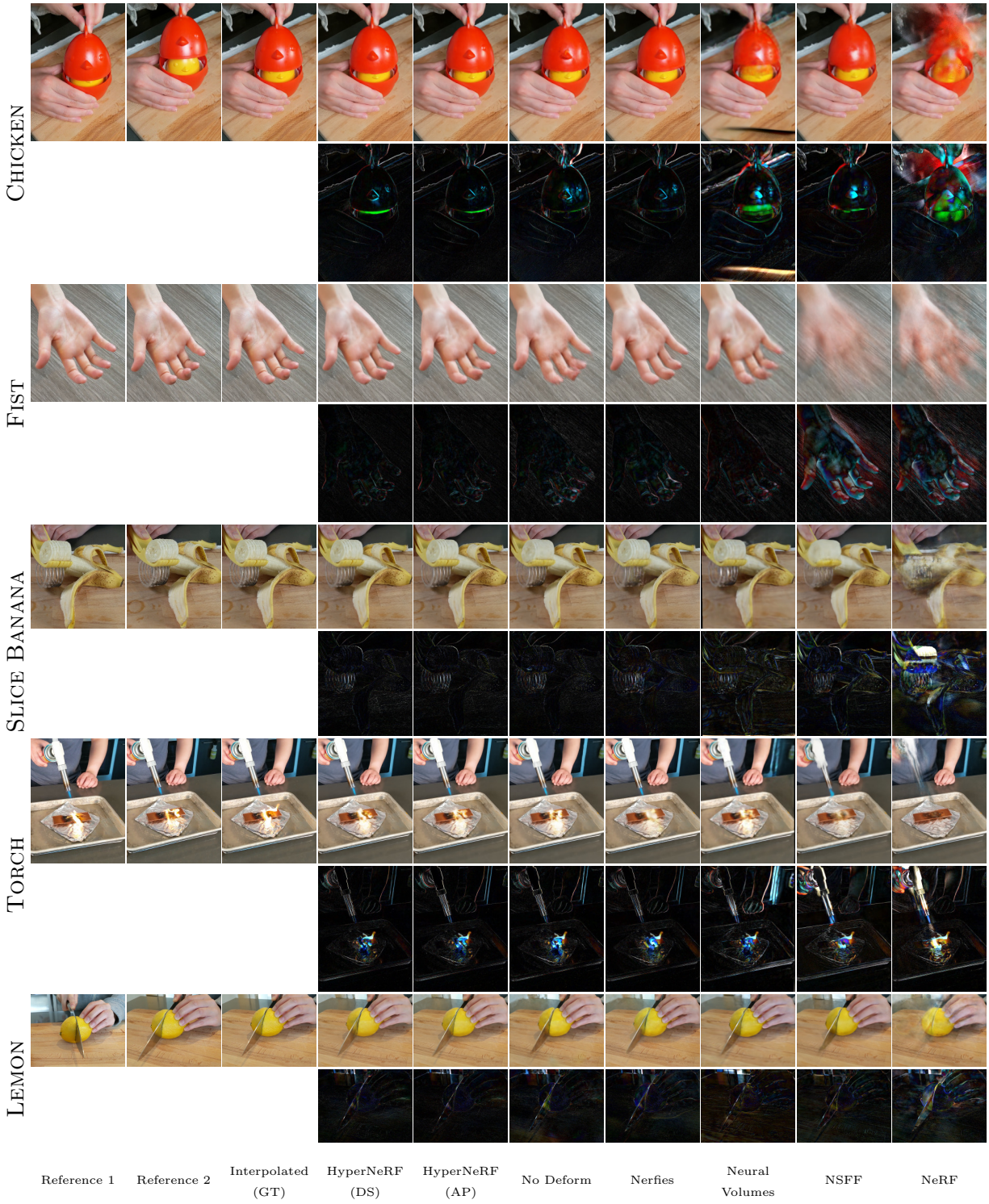


Figure 4.15: (continued) Qualitative comparisons of our method, ablations, and baselines on the interpolation dataset (See §4.4.2). The interpolated frame is rendered by linearly interpolating between the latent codes of the two reference frames. The bottom row of each sequence shows the absolute error. Baselines shown are: Nerfies [159], Neural Volumes [127], NSFF [119], and NeRF [143].

Chapter 5

PHOTOREALISTIC RELIGHTABLE MODELS

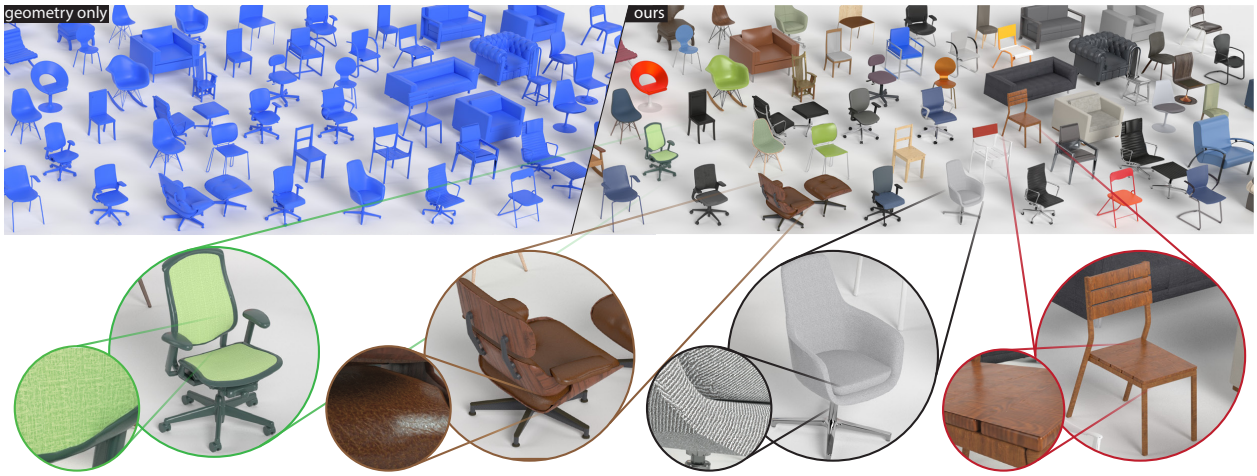


Figure 5.1: Fully automatic texturing of 3D shapes with rich SV-BRDF reflectance models.

So far, we have discussed methods for scene-level capture. But for many applications (e.g., augmented reality), we desire object-level 3D content which can be inserted and blended into virtual environments. Textured 3D models are often used for this purpose, such as in video games or movies. For a 3D model to realistically blend in, it must be *relightable*, meaning each part of the 3D model must be associated with a material model which models how it reflects light. While there are many online repositories containing vast numbers of 3D models, the majority of these do not include this critical material information.

Creating realistic 3D content is quite difficult, and the vast majority of existing models are manually authored. Even more difficult than producing the geometry, which an artist can author using CAD tools, is creating realistic *relightable textures*, as spatially varying reflectance models (SVBRDFs) are 6-dimensional. And while computer vision-based 3D re-

construction research has advanced considerably over the last few years, existing commercial tools produce raw geometry but lack relightable textures and hierarchical part segmentation. As a result, relatively few photorealistic relightable 3D shapes (PhotoShapes) exist, and even fewer are freely available online.

Our goal is to produce thousands of freely available PhotoShapes. To this end, we observe that the problem of creating PhotoShapes can be factored into three more tractable subproblems:

1. we need thousands of good shape models.

2. we need databases of high-quality spatially-varying material models.

3. we need an assignment of materials to shape parts.

The first problem (shape models) is addressed in part by the existence of large model databases like ShapeNet [31] which captures thousands of chairs (and many other categories) spanning myriad shapes and styles. The second problem (appearance models) is addressed in part by existing BRDF/SVBRDF databases and other online material libraries, although many of these are not free (e.g., [2]), and these collections are not as extensive as we would like; we therefore contributed a number of high quality SVBRDFs to round out the collection. The third problem – assignment of shape to materials – is the focus of this chapter.

Given a 3D shape of a chair with a set of parts (e.g., legs, seat, back) and a set of materials (e.g., different types of wood, plastic, leather, fabric, metal) how should we decide which materials to apply to each part? Whereas an artist would choose this assignment manually, we propose to automate this process by leveraging *photos of chairs* on the Internet. I.e., the goal is to use photos of *real* objects as reference to automate the assignment of materials to 3D shapes.

Note that this problem is different than transferring the *texture* of a reference photo to a 3D shape, e.g., [220], as the latter requires generating missing texture for the $> 50\%$ of object surfaces not visible in the photo, and does not enable specular *relighting* (required for

many applications). In addition to solving both of these problems, our approach produces "super-resolution" textures (based on the high-res appearance database) where you can zoom in far beyond the resolution afforded by the reference photo, to see fine wood-grain or stitch patterns up close. The caveat is that these textures are "hallucinated" i.e., they are best matches from the database of textures rather than exact reproductions of the reference object. This means for example, that while the overall look and appearance of the material is often matched well (e.g., "oak", "black leather") and the level of realism is high, the particular knot placement of the wood grain or the texture of the leather may differ significantly from the object in the photo.

Conceptually, we could solve this problem by comparing every reference photo with every 3D shape, textured with every material in the database, and rendered to every viewpoint and with different illuminations. The *good* matches (for each shape part) would yield our desired assignment of materials to shapes. Aside from the obvious scale and combinatorial complexity problems with such an approach, a key challenge is how to robustly *compare* two images where features differ in both shape (e.g., arm height in an office chair) and appearance (e.g., different wood grain). We leverage deep networks, trained on thousands of synthetic renderings of BRDFs applied to 3D shapes to produce robust classifiers that map patches of reference photos to material database instances. We then use fine-scale image alignment techniques and spatial aggregation (CRFs) to assign materials to parts of the shape.

Our fully automatic system is able to produce 2,000 PhotoShapes that very accurately reflect their exemplars, and 9,000 PhotoShapes which deviate slightly but are good representations of their exemplars. In total, our system produces 11,000 photorealistic, relightable 3D shapes.

5.1 Related Work

Material Capture and Representation

A widely used representation for opaque materials is the Bidirectional Reflectance Distribution Function (BRDF) and its spatially varying form (SVBRDF). Estimating BRDF parameters from images is a well studied problem, either in a lab setup [137] or directly from images. The work of [125, 154] optimize for the BRDF parameters given the shape or the illumination respectively. Chadraker etal [29] study how motion cues can assist the estimation. More recent approaches use deep learning to estimate the material parameters [65] from reflectance maps [170], from the image directly [122, 223] or from a RGB-D sequence [103]. For SVBRDFs, [3] introduces a system for easy capture of SVBRDF parameters. [52] infers the diffuse and specular albedo from a moving object. [246] proposes a method for capturing SVBRDFs from a small number of views. The work of [4] uses two photos of the same texture on a flat surface, one taken with a flash and one without. [116] is able to estimate the diffuse albedo, the specularity, and the illumination of a flat texture using a self-augmented neural network.

Diffuse Textures Diffuse textures maps is another technique for material representation. These textures can be directly applied to their corresponding 3D models [46]. The work of [101] uses projective texturing followed by texture synthesis in 3D, after the manual alignment of a 3D object to a 2D image. [86] proposes an approach for detailed geometry and reflectance extraction from single photos using rough 3D proxies. [51] proposes an exemplar based synthesis approach that incorporates 3D cues such as normals and light direction. In a different manner, [106] synthesizes solid textures that were optimized to match the statistics of a 2D image. Apart from their 3D applications, textures have been studied extensively in the image domain. Non-Parametric texture synthesis methods [54, 55] are able to generate plausible textures from small patches. [80] introduced a framework that transfers the texture effects that relate two images to new one. Recently, deep learning approaches have boosted

the quality of the produced textures, synthesizing either from exemplars [62, 182] or semantic labels [35, 90].

Material Recognition

In many computer vision applications it is important to recognize the materials that appear in an image. The method of [183] uses features based on human perception for material classification. [179] proposes a method for discovering attributes suitable for material classification, while [240] identifies materials based on their reflectance. Similarly, [66] classifies materials based on reflectance maps. [13] introduces a large dataset and a deep learning framework for material recognition in the wild (e.g. the Flickr Material Database [184] contains 100 images per class, with the images not beeing representative of everyday scenes). Another dataset for surface material recognition is presented in [234], together with a classification network based on differentiable angular imaging. Moreover, [218] introduces a light-field dataset for material recognition. [41, 42] introduces a texture dataset and a deep learning approach for texture recognition and segmentation, but the texture classes are based on high level attributes.

Image and Shape Dataset Analysis

Wang et al. [221] proposes a method to transfer image segmentation labels to 3D models by aligning the projections of the 3D shapes with annotated images. [87] performs single-view reconstruction by jointly analyzing image and shape collections. Starting from a 3D dataset with material annotations, [93] introduces a method for material suggestions based on material relation of object parts. Similarly, [34] proposes a framework for automatic assignment of materials and textures for indoor scenes based on a set of rules learnt from an annotated database. The work of [91] proposes a framework to infer the geometry of a room from a single image, but the appearance of the 3D models are estimated as the mean diffuse color of the image pixels. [171] aligns 3D models with images to estimate reflectance maps (orientation dependent appearance), and then uses them for shading. Closer to our work

is the method of [220], which transfers textures from images to aligned shapes. However, the transferred textures consist of only a diffuse albedo and need to contain strong patterns. Our work attempts to alleviate this limitation by using rich multi-component representations that capture a large variety of materials.

5.2 Dataset

In this section we describe the three types of datasets that we used in our paper: shape, photo, and texture collections. For this work we have focused on chairs (including a variety of sofas, office chairs, stools etc.). Chairs have a diverse set of appearances and material combinations that make them appealing for our experiments.

5.2.1 3D Shape Collections

The 3D models to be textured come from two free online CAD sources: ShapeNet [31] and Herman Miller [78]. In particular, we used 5,740 3D models from ShapeNet and 90 models from Herman Miller. ShapeNet is a large database of 3D models, containing thousands of 3D models across different categories. The furniture classes that are investigated in this paper are among the most populous, providing a good sampling of the "furniture" geometry. The Herman Miller database contains a small number of 3D models, but with higher quality meshes.

Our 3D models are in OBJ format and are segmented into parts. These parts do not always correspond to semantic groups like "chair leg" but are a byproduct of the 3D shape design process. Some shapes do include material information either as simple Blinn-Phong parameters or as textures, but such materials are usually low quality and inadequate for photorealistic rendering.

Filtering The initial 3D shapes vary in terms of geometric detail and quality, etc. To ensure that the shape collection contains sufficient geometric quality we pre-process the database with the following steps. Firstly, we manually remove the 3D models that they do

not belong to the aforementioned categories. Moreover, we remove unrealistic shapes and shapes with poor quality. Next, we delete vertex doubles and we enable smooth shading. Finally, all the models are resized to fit in a unit bounding cube.

UV Map Generation Most models lack UV mappings which are required for texturing. To estimate the UV maps we use Blender's "Smart UV projection" algorithm [213] for each material segment.

5.2.2 Exemplars: Photographic References for PhotoShapes

We pair 3D shapes with photographic references which we call *exemplars* to guide the appearance of PhotoShapes. Our collection of exemplar consists of of 40,927 product photos that were collected from a) the Herman Miller website, and b) image search engines (Google and Bing), similar to [87]. Specifically, we used 1820 images from Herman Miller and 39107 from the search engines. Product images are suitable for our task because objects of interest are uncluttered and easily segmented from the background (which is usually white). To ensure that images are unique, we remove duplicates by computing dense HOG features [56] on each image and removing images with an L2 distance lower than 0.1. Moreover, a foreground mask is computed for each image using a simple pixel value threshold. The object is then tightly cropped with a square bounding box and resized to 1000x1000.

Figure 5.2: Examples of materials from each class. Rendered with Blender.

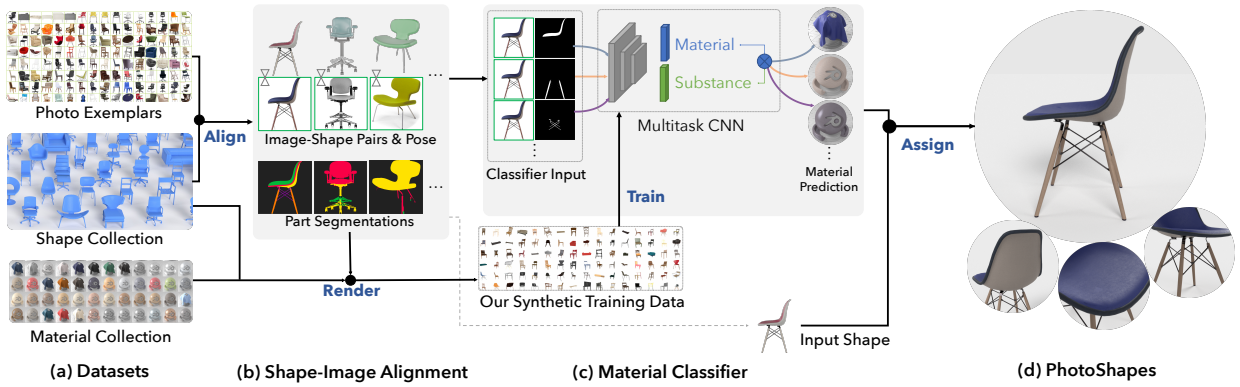


Figure 5.3: An overview of our system. (a) The input to our system is a collection of images, shapes, and materials. (§5.2), (b) we take the shape and image collections and correlate them in an alignment step (§5.3), (c) we take each shape-image pair along with the finely aligned segmentation mask and predict the material of each part (§5.4.2), (d) the output of our system is a large collection of richly textured 3D shapes (novel viewpoints are shown in the figure).

5.2.3 Material Collection

Our goal is to provide realistic, physically based textures to a 3D model collection. Textures are represented as SVBRDFs with spatially varying diffuse, specular and roughness maps. They also contain geometric information via normal and height maps. This representation enables realistic reproduction of materials and seamless incorporation into any modern physically based renderer.

We collected SVBRDFs by scanning real surfaces using [4] and by collecting synthetic textures from online repositories. Specifically, we manually scanned 33 materials in addition to the 34 materials provided in [4]. We downloaded 68 materials from [165], 57 materials from VRay Materials [214] and 238 materials from [2]. We also manually created 15 metals and plastics.

Materials scanned using [4] were converted from their model (similar to BRDF Model A from [23]) to an anisotropic Beckmann model in order to render using Blender. All the other BRDFs were rendered using their designed BRDFs. Poliigon and V-Ray Materials are rendered using the GGX [215] model and Adobe Stock materials are rendered using the

Figure 5.4: The top-5 shape and pose retrievals given an image (outlined).

Figure 5.5: The top-5 image retrievals (outlined) given a shape. Computed using a reverse-index.

Disney Principled BRDF [27]. In total, our database consist of 48 leathers, 154 fabrics, 105 woods, 86 metals, and 60 plastics. Examples of the materials are shown in Figure 5.2.

Normalizing Scale Materials are scanned or created with an arbitrary and unknown scale. In order to use materials consistently, we manually assign a scale value s_i for each material $m_i \in \mathcal{M}$. This value is used as a scaling factor for the UV mappings which are scaled by a factor of $\log s_i$ during rendering.

Environment Maps We also have a small set of 30 HDR environment maps from [46], [239], and [2]. We select environment maps that simulate studio-like lighting conditions as use them for all of our renderings.

5.3 Shape-Image Alignment

Given a collection of uncorrelated 3D shapes and images of the same category, we wish to synthesize realistic textured versions of each 3D shape. To achieve this, we propose a system to extract appearance information from the images and transfer that information onto the shape collection. We pose the problem as a classification problem in which the goal is to assign a material model from our texture dataset to each 3D shape part. Our system is comprised of two parts:

1. *Coarse step*: assigns to each shape a list of exemplars and associated camera poses

2. *Fine step*: creates a pixel-wise alignment between shapes and exemplars

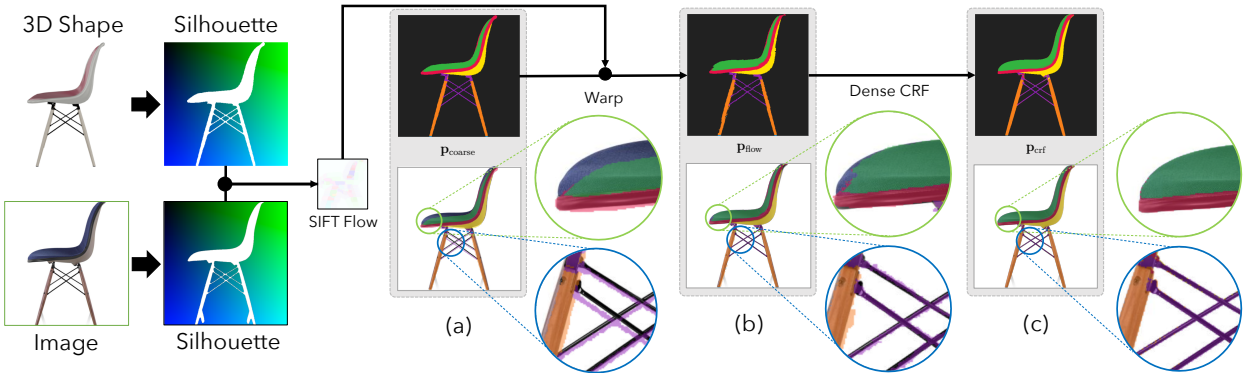


Figure 5.6: We refine a coarsely aligned mesh segment ID map to better align with the image. The shape segmentation map projected onto the image (a) after coarse alignment; (b) after applying the flow field computed using SIFT Flow; (c) after applying our dense CRF.

We use the following terminology throughout the paper: a *shape* is a 3D model obtained from an online shape collection. Each shape is by construction divided into *object parts* defining structural divisions (seat, arm, leg, etc.) and *material parts* ($\mathcal{P}$) defining which objects should share the same material.

In order to texture a 3D shape, we refer to a set of associated image exemplars and use them as a proxy for reasoning about plausible textures for the shape. For this to be possible we must first compute an association between the collection of 3D shapes and exemplar images. We call this task *alignment* and break it down into two steps: 1) a *coarse* step, and 2) a *fine* step.

5.3.1 Coarse Step: Shape to exemplar matching.

We seek a list of exemplars for each 3D shape, as well as the camera pose for every such shape-exemplar pair. We pose coarse alignment as an image retrieval problem, solving shape retrieval and pose estimation simultaneously. Similar approaches are taken in [87] and [220]. For efficiency, we solve this by creating a reverse-index from exemplars to the top k shape renderings. Inverting this index gives us our desired output.

We render each shape from various viewpoints sampled from a sphere around the object.

The camera is parameterized in spherical coordinates. 50 elevation values are uniformly sampled in $\phi \in [\frac{\pi}{4}, \frac{9}{16}\pi]$ and $(50 \sin \phi)$ azimuth values sampled uniformly over $\theta \in [0, 2\pi)$. This results in 456 distinct viewpoints.

Distance Metric We require a distance metric in order to compare the compatibility of a rendering and an image. The alignment problem is then

$$M(I) = \underset{\phi \in \Phi, \theta \in \Theta, s \in |\mathcal{S}|}{\arg\min} \|F(R(s; \phi, \theta)) - F(I)\| \tag{5.1}$$

where M is a function that returns the top match, I is the image query, R is the renderer, $\mathcal{S}$ is the set of all shapes, Φ is the set of all elevations, Θ is the set of all azimuths, and F is a feature descriptor.

We use the HOG descriptor from [56] as F. This feature descriptor has the benefit of low dimensionality making computation and comparisons extremely efficient. We compute descriptors of size 100x100 with a bin size of 8 yielding 1352 dimensional features. The input image is blurred with a Gaussian filter with $\sigma = 0.1$ in order to reduce texture effects.

During comparison, the rendering $R(s; \phi, \theta)$ and the image I are both cropped with a square bounding box around their foreground masks. This allows us to perform the coarse alignment with a simple spherical coordinate camera model forgoing focal length or translation parameters.

5.3.2 Fine Step: Segmentation Refinement

3D shapes are segmented into object and material parts by their authors upon construction. We assume that any parts of the shape which have the same material label share the same apperance. One may also use the object parts as supervision for this purpose; however, we find that these tend to be over segmented and lack the symmetry found in material segments (e.g. each leg of a chair may be assigned a different material). Given the coarse alignment we can compute a 2D material part labeling $\mathbf{p}_{\text{coarse}} \in \mathcal{P}^N$ for an image of size N by projecting

the shape parts $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{P}|}\}$ with the estimated camera pose. The effect of this is shown in Figure 5.6(a).

A naive projecting of the coarsely aligned part mask is insufficient for associating the two modalities. The mask does not perfectly align with the exemplar and thin structures such as chair legs may have zero overlap. We use the coarse alignment as initialization and perform an additional refinement step in order to get a cleaner pixel-wise alignment of the projected part mask.

SIFT Flow We use the approach from [220] in which we compute a flow which warps the projected shape segment map onto the exemplar. The flow is computed by using the SIFT Flow algorithm [121] on the silhouettes of each map. We encode the vertical and horizontal pixel coordinates into the blue and green pixels of the silhouette image (as in Figure 5.3). This prevents the SIFT Flow step from overly distorting the mask. The resulting warped mask $\mathbf{p}_{\text{flow}} \in \mathcal{P}^N$ is shown in Figure 5.6(b).

Dense CRF The SIFT Flow refinement results in an overlapping but noisy segmentation. We further clean the segmentation mask by using a dense CRF [108] in the same manner as [13] (Please see supplemental materials for details). The resulting part mask $\mathbf{p}_{\text{crf}} \in \mathcal{P}^N$ which we use for the rest of the system is shown in Figure 5.6(c). The aligned part mask enabled us to share information between shapes and corresponding image exemplars.

5.3.3 Substance Segmentation

We first use the aligned image exemplar to infer *types* of materials, a.k.a. "*substances*" for each part of the aligned object. In the next section, we will convert these substances into fine-grained SVBRDFs. We segment the image and label each pixel with a substance category using [13]. For our experiments we use the substances 'leather', 'fabric', 'metal', 'wood', and 'plastic'. Similar categories are mapped to a canonical category (e.g. 'carpet' to 'fabric'). All other category probabilities are set to zero and the remainder are re-scaled

to sum to 1.0. This process results in a substance mask $\mathbf{q} \in \mathcal{Q}^N$ where $\mathcal{Q} = \{q_1, q_2, \ldots, q_{|\mathcal{Q}|}\}$ is the set of substance labels. We compute a substance labeling of the shape $\mathbf{q}_{\text{shape}} \in \mathcal{Q}^{|\mathcal{P}|}$ by choosing the substance label that has the most overlap. Let s_i be the substance label of part i. This process may also yield a cleaner substance segmentation of the image. See supplementary material for an example.

This process assigns a substance label to each aligned 3D shape part computed from §5.3.2.

5.4 Image Segment to SVBRDF

Our objective is to assign a plausible SVBRDF to each part of a 3D shape. One approach is to extract planar patches and optimize a texture as in [220] with an SVBRDF regression method such as [116]. However, we find that extracting patches from images yields low resolution, distorted textures which are difficult to analyze. Extracting local planar patches also loses global context which is useful for inferring glossiness.

We instead tackle this problem as a classification problem. The input is an image and a corresponding binary mask representing a single material. The output is a material label $m \in \mathcal{M}$ chosen from our collection of SVBRDFs. Ideally we would have a collection of real images with ground truth SVBRDF labelings. In practice, it is difficult to define such a task. We also found human judgment of reflectance (as in [12]) to be noisy and low quality. We therefore generate synthetic data where we know ground truth.

5.4.1 Synthesizing Training Data

Synthetic data has shown to be effective for training or augmenting models that generalize to real world applications [172, 195]. We therefore sidestep the difficulty of collecting ground truth by creating our own. Given our 3D shape and material databases, we can create a large amount of training data by applying different materials to shapes and rendering to a range of camera viewpoints under different illuminations.

Camera Pose Prior We find that there is a strong bias in camera poses in real images (e.g., chairs are rarely photographed from below). We thus uniformly sample from the distribution of camera poses obtained in the coarse alignment step.

Substance Prior Substances do not occur randomly in objects. Legs of chairs are usually not made of leather and a sofa is usually not upholstered with metal. We leverage the shape substance labelings $q_{p_i} \in \mathbf{q}_{shape}$ in §5.3.3 to enforce a substance prior. Instead of selecting a completely random material, we condition on the substance category and sample $m_{p_i} \sim U(\{m|m \in \mathcal{M}, q_m = q_{p_i}\})$.

Texture Scale Normalization Different tessellation and UV mappings can arbitrarily change the rendered scale of our textures. We normalize the UV scale for each mesh segment S_i by computing a density $D_i = A_i^{\text{uv}}/A_i^{\text{world}}$ where A_i^{uv} is the local UV-space surface area of the mesh and A_i^{world} is the local world-space surface area. The UV coordinates for the segment is then scaled by $\frac{1}{D_i}$. This method assumes little or no distortion in the UV mapping.

Randomized Rendering To generate a single random rendering, we uniformly sample a shape-exemplar pair computed in §5.3.1. Given a pair, we 1) sample a camera pose from the distribution computed above, 2) assign a random material (SVBRDF) to each shape part conditioned on the substance label computed in §5.3.3, and 3) select a random environment map.

To improve classifier robustness, we further augment our data as follows: 1) Randomly jitter azimuth and elevation by $\Delta\theta \sim U([-\frac{\pi}{12}, \frac{\pi}{12}])$ and $\Delta\phi \sim U([-\frac{\pi}{24}, \frac{\pi}{24}])$, 2) randomly select a field of view $\text{fov}_x \sim U([50, 60])$, 3) randomly select a camera distance $r \sim U([1.3, 1.75])$, 4) randomly scale the radiance of the environment map by $s_{\text{env}} \sim U([0.9, 1.2])$, 5) randomly scale, rotate, and translate UV mappings by ($\Delta s_{\text{uv}} \in [-1, 0.5]$, $\Delta\theta_s \in [0, 2\pi]$, and $\Delta x, \Delta y \in [0, 1]^2$) respectively. We use Blender to generate 156,262 renderings (examples in supplementary material).

5.4.2 Material Classification

Our synthetic dataset is generated by conditioning on *substances*. We would like to be able to generate PhotoShapes with more accurate fine grained *materials* (e.g., a specific 'oak', 'cherry', 'maple', instead of just 'wood'). Although material assignments were only conditioned on substance categories, the renderings of our synthetic dataset from §5.4.1 contain ground truth labels for which specific *material* is rendered at each pixel.

We directly use the renderings and their ground truth *materials* to train a classifier which predicts which materials are present in a specified image. We experimented with other methods such as color histogram matching but found that such brute-force matching approaches are not practical when comparing a large number of materials. Our classifier is a feed-forward neural network which is efficient even for a large number of materials.

The input to our classifier is an image exemplar concatenated with a binary segmentation mask. The input mask represents the portion of the image that is to be classified. For example, if the mask had non-zero- values only within the fabric upholstery (as in Figure 5.3(c)) the desired output would be a blue fabric. The output of our classifier is an $|\mathcal{M}|$ dimensional vector $\mathbf{x}^m$ which when applied a soft-max operator becomes a discrete probability mass. We optimize this class labeling using a cross entropy loss

$$\mathcal{L}_{\text{mat}}(\mathbf{x}^m, y_i^m) = -\log\left(\frac{\exp x_i^m}{\sum_j \exp x_j^m}\right)$$

where y_i^m is the ground truth label.

Multitask Learning

We find that naively training a network only on material class information generalizes poorly to real images. Intuitively this makes sense since we have given the system no information about the relative affinity between different materials e.g., it is less wrong to classify beech wood as cherry wood than it is to classify it as a black leather. We introduce an auxiliary task to our network in order to regularize our feature space and to teach the network the relative affinity between materials.

Substance Classification Mis-classifying a wood material as leather is more detrimental than classifying it as a different wood. As such, we add an additional fully connected layer to our network which predicts the *substance* category $q \in \mathcal{Q}$ of the input (wood, leather, plastic, etc.) The ground truth labels come directly from annotations of our material dataset. The substance is also a classification and is optimized using a cross entropy loss in a similar fashion to the material loss:

$$\mathcal{L}_{\text{sub}}(\mathbf{x}^s, y_i^s) = -\log\left(\frac{\exp x_i^s}{\sum_j \exp x_j^s}\right)$$

where y_i^s the ground truth label.

Combining Losses The most straightforward way to optimize our objective is to compute a weighted sum of our loss functions:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \lambda) = \mathcal{L}_{\text{mat}}(\mathbf{x}^m, \mathbf{y}^m) + \lambda \mathcal{L}_{\text{sub}}(\mathbf{x}^s, \mathbf{y}^s)$$

for some weighting term λ. However, we found it more efficient to use the uncertainty weighted multitask loss from [100] which defines a weighting based on learned homoscedastic (task-dependent and not data-dependent) uncertainties $\hat{\sigma}_m^2, \hat{\sigma}_s^2$. Concretely, our loss function is formulated as:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \hat{\sigma}) = \mathcal{L}_{\text{mat}}(\mathbf{x}^m, \mathbf{y}^m)\hat{\sigma}_m^{-2} + \log \hat{\sigma}_m^2 \tag{5.2}$$

$$+ \mathcal{L}_{\text{sub}}(\mathbf{x}^s, \mathbf{y}^s)\hat{\sigma}_s^{-2} + \log \hat{\sigma}_s^2 \tag{5.3}$$

In practice, we optimize for the log variance $\hat{s} := \log \hat{\sigma}^2$ as it avoids a possible divide by zero and is more numerically stable:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \hat{\mathbf{s}}) = \mathcal{L}_{\text{mat}}(\mathbf{x}^m, \mathbf{y}^m)\exp(-\hat{s}_m) + \hat{s}_m \tag{5.4}$$

$$+ \mathcal{L}_{\text{sub}}(\mathbf{x}^s, \mathbf{y}^s)\exp(-\hat{s}_s) + \hat{s}_s \tag{5.5}$$

We initialize $\hat{s}_m = 0.0, \hat{s}_m = -1.0$.

Model Architecture and Training

We use Resnet-34 [73] with weights initialized to those pre-trained on ImageNet. We add a 4th alpha channel to the input layer of the network which represents the segment of the image we wish to classify. The associated filters of the alpha channel are initialized with a random Gaussian. The output of the network is a score for each of the $C + 1$ categories mapping to a material and a background class.

We train the network with stochastic gradient descent (SGD) with a fixed learning rate of 0.0001 until convergence (about 100 epochs). We perform standard data augmentations: random rotations, random crops, random aspect ratio changes, random horizontal flips, and random chromatic/brightness/contrast shifts.

Pretraining on Real Images

Although our network trained only on synthetic renderings generalizes fairly well to real images, we found that having natural image supervision helps the network learn a more robust model. We pre-train our network on the dataset of [13] using their ground truth region polygons to generate input masks. Our data mostly has white backgrounds and we therefore interleave whole image inputs with cropped inputs with white backgrounds.

We randomly split the OpenSurfaces dataset into training and validation sets at a ratio of 9:1. We fine-tune the network initialized to weights pretrained on ImageNet [49] and train using only the substance task with a learning rate of 0.0001 until convergence. The network reaches 86.77% top-1 validation precision on the our pretraining validation set.

Inference

Given an image we take the segmentation mask computed in §5.3.2 and infer a material for each segment. We find that we are able to improve material prediction performance by weighting the material prediction by the confidence of the corresponding substance category prediction.

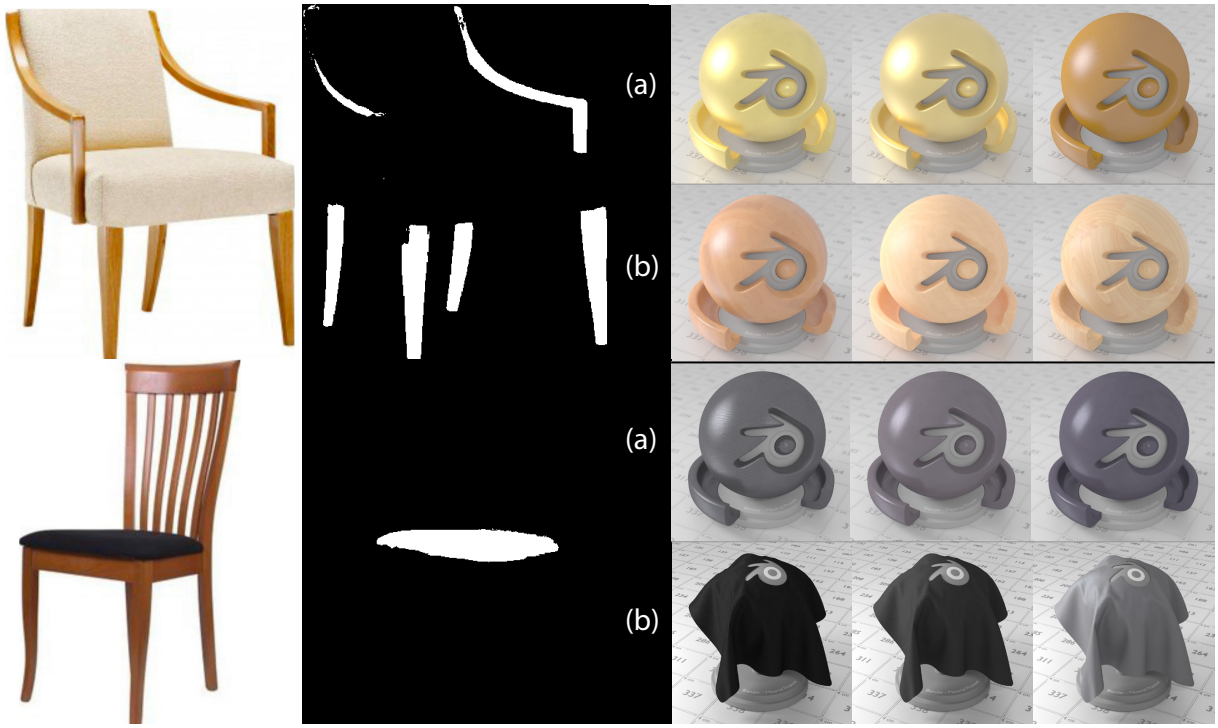


Figure 5.7: The top-3 material predictions of our material classification network for the input shown on the left. (a) shows predictions when trained without substance supervision, (b) shows predictions when trained with substance supervision.

5.4.3 Generating PhotoShapes

The final objective is to create a collection of photorealistic, relightable 3D shapes. Consider the collection of all aligned shape-exemplar pairs computed in §5.3 as PhotoShapes *candidates*. These candidates become PhotoShapes when each of their material parts are assigned a relightable texture. The latter step is done simply by applying §5.4.2 to each photo-aligned material part (using the mask from §5.3.2). For our experiments, we took the top 12 matching exemplars for each shape (ordered by HOG distance) and discarded any pairs with a HOG distance < 8. We then applied our material classifier to the remaining candidates. This process yields 29,133 PhotoShapes. Examples are in Figure **??**.

5.5 *Experiments*

In this section, we describe implementation details, evaluations, comparisons to prior work, limitations, and applications.

5.5.1 Baselines

Built-in Textures The simplest baseline is to compare with the materials that come by default from each shape model. Most of the shapes in our dataset lack any meaningful material properties. Besides the very few with high quality textures, most shapes have either arbitrary colors or low resolution textures.

No Material Classification We evaluate a version of our method without the material classification step. We assign random materials given the substance computed in §5.3.3. This is akin to our training data generation process.

Color Matching A naive method of matching materials is to render a shape with all possible materials and compare the resulting color distribution with the exemplar image. Our experiments show that such brute-force approaches were too slow for large-scale applications (rendering thousands of object with hundreds of materials takes a long time). Such comparisons also have difficulty accounting for complicated textures and non-uniform lighting effects such as shadows and specularities. We show a comparison in Figure 5.8.

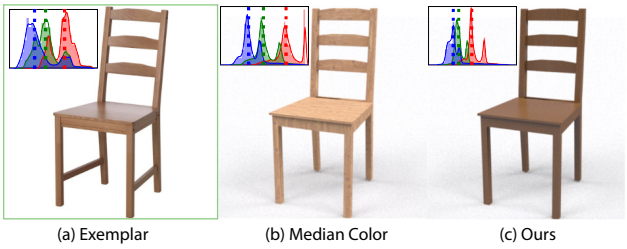


Figure 5.8: Matching by median color fails to account for specular reflections and fetches a diffuse, lighter material. We include the color histogram of each image on the upper left with the median color (dotted lines).

Projective Texturing Another approach for appearance modeling is projective texturing. If the alignment between the 3D model and the image is good [46, 101], the image can be

Figure 5.9: Our method is able to produce different plausible materials (bottom) of the same Shape (left) given different exemplar images (top outlined).

used as a texture for the 3D model. However, good alignment is a very difficult task, even with manual intervention. An additional approach that models the appearance of an object as "baked" material and illumination properties is reflectance maps [171]. Figure 5.13 shows the results of projective texturing and reflectance map shading compared to our approach.

5.5.2 Results

We show a sample of our results in Figure 5.15, comparing to the built-in materials, our results with no classifier, and our full results. We also show the benefit of using high-resolution relightable textures in Figure 5.12.

We also show results for categories other than chairs in Figure 5.10. Note that these results are produced without additional training. Results may be improved by adding relevant materials (e.g., rubber for tires) and further training the classifier.

5.5.3 User Study

We conducted a user study on Amazon Mechanical Turk to evaluate the performance of our method. We showed users an image and asked them to specify whether they thought the image was a "real photograph" or "generated by a computer". We tested our method with three different baselines:

Table 5.1: User Study. The ratio of users who thought our results were real photographs.

	ShapeNet		Herman Miller	
	Real	Fake	Real	Fake
Built-in	32%	68%	37%	63%
Ours (No Classifier)	41%	59%	43%	57%
Ours	**47%**	53%	**51%**	49%
Photographs	81%	19%	83%	17%

Figure 5.10: Without any additional training, our pipeline produces plausible results for motorcycles (top left), pillows (top right), coffee tables (bottom left) and cabinets (bottom right). A green outline indicates the exemplar followed by our result to the right. The motorcycle tires are assigned metallic and leather materials as we lack a rubber material in our dataset. The knobs of the cabinet were missed by the fine alignment step due to the small size and sharp color variation of the chrome.

built-in textures, our pipeline but without

the material classifier, our full method, and real images. We performed the study on the ShapeNet and Herman Miller datasets separately. For ShapeNet results, we sampled 1000 result renderings uniformly at random. For Herman Miller results we sampled 500. Results are shown in Tab. 5.1.

5.5.4 Shape to PhotoShapes Conversion Rate

We also manually evaluated our results. We categorize resulting PhotoShapes from our pipeline into the following categories: good, acceptable, and failure. *Good* results represent their exemplar image almost exactly. *Acceptable* results have slight differences from the exemplar but are good representations.

Figure 5.11: Representative examples of our categorization of results. (a) Good models are good representations of the original exemplar. (b) Acceptable models have slight differences but are overall plausible. (c) Failures, for which we identify the following modes: (i) material mis-classification, (ii) material mis-classification caused by ambiguous appearance (plastic sometimes looks like leather etc.), (iii) color mismatch, (iv) over-segmented mesh causing material discontinuity, (v) under-segmented mesh making it impossible to assign correct materials, and (vi) mis-alignment of the exemplar and shape (includes retrieving the wrong object)

Table 5.2: Our generated PhotoShapes divided into good, acceptable, and failure categories. We also show the sum of the good and acceptable classes. See Tab. 5.3 for a more detailed division of failures.

	Good	Acceptable	Failure	Good+Acc
ShapeNet	6.08%	32.76%	61.15%	38.85%
Herman Miller	37.45%	30.45%	32.10%	67.90%

We ignore shapes with low mesh quality (very low polygon count, holes, incorrect normals) and bad exemplars (watermarks, transparent materials, etc.). Everything else are considered *failures*. We identify four main failure modes: (1) material mis-classification e.g., metal instead of wood, (2) color or pattern mismatch, (3) failures due to the under- or over-segmentation of meshes, and (4) incorrect shape retrievals or mis-alignments. Representative examples of each of each category are shown in Figure 5.11.

We evaluate our two input shapes collections (ShapeNet and Herman Miller) separately. We generate 28,432 PhotoShapes for ShapeNet and 701 PhotoShapes for Herman Miller shapes. Since manually sorting all results is not feasible, we evaluate quality on a random sampling. We uniformly randomly sample 1,322 PhotoShapes from ShapeNet and 243 Pho-

toShapes from Herman Miller and categorize them into the aforementioned classes. We found most failure cases were due to mis-alignments and under-segmentations of the mesh. Tab. 5.2 shows the division between each category, and Tab. 5.3 shows a finer division between different failure modes. Our shapes from Herman Miller have considerably higher mesh quality, resulting in a higher success rate due to a lower number of mis-alignments.

Extrapolating from our categorization, our system was able to produce around 2,100 'good' PhotoShapes for ShapeNet and 262 'good' PhotoShapes for Herman Miller. Including 'acceptable' results, we generated 11,000 PhotoShapes for ShapeNet and 475 PhotoShapes for Herman Miller.

We also report input shape coverage (projected numbers shown in parentheses). Our annotations show that 14.93% (856) of ShapeNet shapes and 68.42% (69) of Herman Miller shapes had least one 'good' PhotoShapes. When we include 'acceptable' results, ShapeNet had a success rate of 64.25% (3,687) and 91.23% (92) for Herman Miller.

Table 5.3: Different modes of failure. 'align' refers to mis-alignment errors, 'subst' refers to substance misclassification errors, 'color' refers to incorrect color or pattern predictions, 'und-seg' refers to errors due to the under-segmentation of the shape, and 'ov-seg' refers to awkward results due to models being oversegmented. See Figure 5.11 for examples.

	align	subst	color	und-seg	ov-seg
ShapeNet	40.18%	27.93%	11.86%	17.98%	2.04%
Herman Miller	14.10%	38.46%	32.05%	14.10%	1.28%

While a success rate like 64% may not sound impressive, it is very significant in the context of the problem that we're trying to solve, i.e., generating a large dataset of high quality PhotoShapes. I.e., **we have generated over 3,500 photorealistic, relightable, 3D shapes of chairs**. It's not critical that we texture every shape, as many of these (particularly with ShapeNet) were artist-created and may not correspond to real furniture for which photo exemplars exist.

5.5.5 Material Classifier Evaluation

Data Split We split our synthetic rendering dataset into a training set and a validation set. Since our materials are our prediction categories, we perform the split on the set of shapes and environment maps that are used to generate our renderings. We set aside 10% of

our shapes and environment maps as validation and use the rest for training. This process yields 156,511 renderings for training and 15,872 renderings for validation.

Ablation Study We evaluate our network with the following metrics: a) the top-1 material class precision, b) top-5 material class precision, c) top-1 substance class precision of the substance output layer, and d) the top-1 substance class precision of the substance predicted by aggregating the confidence of each material class by substance category.

We compare four different versions of our network: 1) trained only with material class supervision, 2) trained with material and substance class supervision, 3) pretrained on OpenSurfaces and then trained only with material class supervision, and 4) pretrained on OpenSurfaces and then trained with material and substance class supervision. The results for these metrics are shown in Tab. 5.4.

Table 5.4: Ablation Study. We compare different versions of our model on our synthetic validation set. We try the permutations of whether or not we pretrain on OpenSurfaces and the presence of an additional substance task. (a) mtl@1 is the top-1 validation precision, (b) mtl@5 is the top-5 precision, (c) sub@1 is the top-1 substance precision of the substance task layer prediction, and (d) sub-mtl@1 is the top-1 precision of the substance prediction implied by the material prediction.

pretrain	sub	mtl@1	mtl@5	sub@1	sub-mtl@1
N	N	33.87%	61.95%	-	71.38%
N	Y	37.17%	64.31%	75.50%	76.58%
Y	N	**37.59%**	64.69%	-	72.37%
Y	Y	37.34%	**65.45%**	**75.51%**	**76.60%**

Our additional substance categorization task significantly boosts the validation accuracy of our classifier. We also find that material predictions are qualitatively more robust when trained with substance supervision as shown in Figure 5.7.

5.5.6 Image-Based SVBRDF Retrieval

Finding or designing a suitable material for a 3D scene may be time consuming. An application of our work is retrieving a BRDF based on an image. A user-specified region of an image may be used as input to our classifier in order to produce a ranking of BRDFs.

Despite being provided very little natural image supervision (only pretraining on Open-

Surfaces), our material classifier is able to generalize surprisingly well to general photographs that do not have white backgrounds. Examples of predictions on such images are shown in Figure 5.14.

5.6 Discussion

5.6.1 Limitations and Future Work

Our results are limited in part by the variety of materials available – we cannot reconstruct the wheels of the motorcycle in Figure 5.10 because we do not have a rubber tire material. Our results could be improved by expanding the material database or by exploring methods to augment the current materials. This may be done by synthetically varying color and glossiness, or by *synthesis* of novel SVBRDFs.

Our work relies heavily on reliable matching of photos and shapes. Most of our failures come from mis-alignments or under-segmentations of the input shape. Adding more exemplar images and filtering low-quality shapes would yield better results. Incorporating more sophisticated alignment and segmentation methods are interesting topics for future work.

5.6.2 Conclusion

We have presented a framework that assigns high quality relightable textures to a collection of 3D models with limited material information. The textures come from a large database and the material-to-3D assignment is performed with the guidance of real images to ensure plausible material configurations, and yields thousands of high quality PhotoShapes.

Figure 5.12: Here we show some of our results and close-ups. By using SVBRDFs to model we are able to infer appearance at great detail, even if the exemplar image has low resolution.

Figure 5.13: Comparisons of producing a novel view of an exemplar image using projective texturing, reflectance maps, and then our method.

Figure 5.14: Without any additional training, our network is able to make reasonable predictions on natural images. The input image and colored outline for the segmentation mask is shown on the left. The top BRDF retrieval for each segment are shown on the right.

Figure 5.15: Selected Results. (a) shows the input shape, (b) the exemplar image, (c) a rendering with the default materials that come with the shape, (d) rendering with materials sampled conditioned on substance category, (e) renderings of our final PhotoShapes. Please see supplementary materials for more results.

Chapter 6

NEURAL RENDERING FOR ZERO-SHOT POSE ESTIMATION

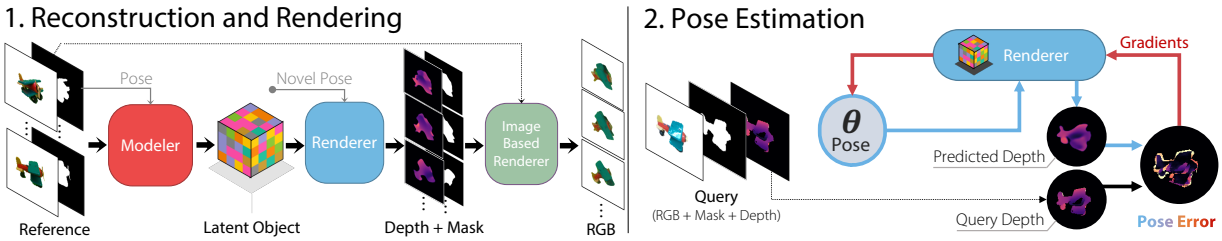


Figure 6.1: We present an end-to-end differentiable reconstruction and rendering pipeline. We use this pipeline to perform pose estimation on unseen objects using simple gradient updates in a render-and-compare fashion.

Nerfies, presented in Chapter 3, and HyperNeRF, presented in Chapter 4 are examples of *neural rendering* methods. Neural rendering methods are usually differentiable, meaning that they can be "inverted" by performing gradient descent on input parameters with respect to their outputs. In this chapter, I introduce LatentFusion [158], a framework for performing both 3D reconstruction and rendering using a neural network. This neural network takes posed images of an object as input and can render it from any novel viewpoint. I show how we can exploit this differentiable pipeline to solve zero-shot pose estimation.

LatentFusion addresses 6D object pose estimation. An object pose is typically defined by a 3D orientation (rotation) and translation comprising six degrees of freedom (6D). Knowing the pose of an object is crucial for any application that involves interacting with real world objects. For example, in order for a robot to manipulate objects it must be able to reason about the pose of the object. In augmented reality, 6D pose estimation enables virtual interaction and re-rendering of real world objects.

In order to estimate the 6D pose of objects, current state-of-the-art methods [50, 216, 231]

require a 3D model for each object. Methods based on renderings [208] usually need high quality 3D models typically obtained using 3D scanning devices. Although modern 3D reconstruction and scanning techniques such as [148] can generate 3D models of objects, they typically require significant effort. It is easy to see how building a 3D model for every object is an infeasible task.

Furthermore, existing pose estimation methods require extensive training under different lighting conditions and occlusions. For methods that train a single network for multiple objects [231], the pose estimation accuracy drops significantly with the increase in the number of objects. This is due to large variation of object appearances depending on the pose. To remedy this mode of degradation, some approaches train a separate network for each object [50, 199, 208]. This approach is not scalable to a large number of objects. Regardless of using a single or multiple networks, all model-based methods require extensive training for unseen test objects that are not in the training set.

We investigate the problem of constructing a 3D object representations for 6D object pose estimation without 3D models and without extra training for unseen objects during test time. The core of our method is a novel neural network that takes a set of reference RGB images of a target object with known poses, and internally builds a 3D representation of the object. Using the 3D representation, the network is able to render arbitrary views of the object. To estimate object pose, the network compares the input image with its rendered images in a gradient descent fashion to search for the best pose where the rendered image matches the input image. Applying the network to an unseen object only requires collecting views with registered poses using traditional techniques [148] and feeding a small subset of those views with the associated poses to the network, instead of training for the new object which takes time and computational resources.

Our network design is inspired by space carving [110]. We build a 3D voxel representation of an object by computing 2D latent features and projecting them to a canonical 3D voxel space using a *de*projection unit inspired by [150]. This operation can be interpreted as space carving in latent space. Rendering a novel view is conducted by rotating the latent

voxel representation to the new view and projecting it into the 2D image space. Using the projected latent features, a decoder generates a new view image by first predicting the depth map of the object at the query view and then assigning color for each pixel by combining corresponding pixel values at different reference views.

To reconstruct and render unseen objects, we train the network on the ShapeNet dataset [30] randomly textured with images from the MS-COCO dataset [120] under random lighting conditions. Our experiments show that the network generalizes to novel object categories and instances. For pose estimation, we assume that the object of interest is segmented with a generic object instance segmentation method such as [232]. The pose of the object is estimated by finding a 6D pose that minimizes the difference between a predicted rendering and the input image. Since our network is a differentiable renderer, we optimize by directly computing the gradients of the loss with respect to the object pose. Fig. 6.1 illustrates our reconstruction and pose estimation pipeline.

Some key benefits of our method are:

1. *Ease of Capture* – we perform pose estimation given just a few reference images rather than 3D scans;

2. *Robustness to Appearance* – we create a latent representation from images rather than relying on a 3D model with baked appearance; and

3. *Practicality* – our zero-shot formulation requires only one neural network model for all objects and requires no training for novel objects.

In addition, we introduce the Model-free Object Pose Estimation Dataset (MOPED) for evaluating pose estimation in a zero-shot setting. Existing pose estimation benchmarks provide 3D models and rendered training images sequences, but typically do not provide casual real-world reference images. MOPED provides registered reference and test images for evaluating pose estimation in a zero-shot setting.

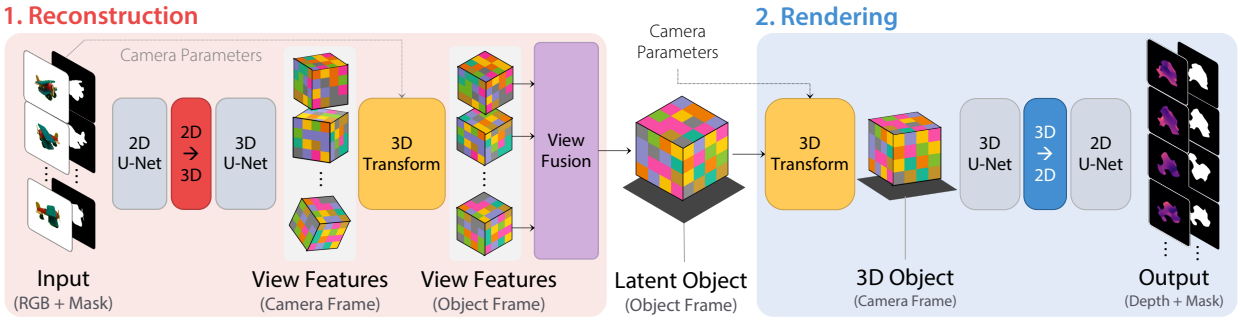


Figure 6.2: A high level overview of our architecture. 1) Our modeling network takes an image and mask and predicts a feature volume for each input view. The predicted feature volumes are then fused into a single canonical *latent object* by the fusion module. 2) Given the latent object, our rendering network produces a depth map and a mask for any camera pose.

6.1 Related Work

Pose Estimation Pose estimation methods fall into three major categories. The first category tackles the problem of pose estimation by designing network architectures that facilitate pose estimation [99, 109, 145]. The second category formulates the pose estimation by predicting a set of 2D image features, such as the projection of 3D box corners [85, 164, 208, 216] and direction of the center of the object [231], then recovering the pose of the object using the predictions. The third category estimates the pose of objects by aligning the rendering of the 3D model to the image. DeepIM [117] trains a neural network to align the 3D model of the object to the image. Another approach is to learn a model that can reconstruct the object with different poses [50, 199]. These methods then use the latent representation of the object to estimate the pose. A limitation of this line of work is that they need to train separate auto-encoders for each object category and there is a lack of knowledge transfer *between* object categories. In addition, these methods require high-fidelity textured 3D models for each object which are not trivial to build in practice since it involves specialized hardware [187]. Our method addresses these limitations: our method works with a set of reference views with registered poses instead of a 3D model. Without additional training, our system builds a latent representation from the reference views which

can be rendered to color and depth for arbitrary viewpoints. Similar to [50, 199], we seek to find a pose that minimizes the difference in latent space between the query object and the test image. Differentiable mesh renderers have been explored for pose estimation [39, 155] but still require 3D models leaving the acquisition problem unsolved.

3D shape learning and novel view synthesis Inferring shapes of objects at the category level has recently gained a lot of attention. Shape geometry has been represented as voxels [40], Signed Distance Functions (SDFs) [156], point clouds [235], and as implicit functions encoded by a neural network [190]. These methods are trained at the category level and can only represent different instances within the categories they were trained on. In addition, these models only capture the *shape* of the object and do not model the *appearance* of the object. To overcome this limitation, recent works [150, 151, 190] decode appearance from neural 3D latent representations that respect projective geometry, generalizing well to novel viewpoints. Novel views are generated by transforming the latent representation in 3D and projecting it to 2D. A decoder then generates a novel view from the projected features. Some methods find a nearest neighbor shape proxy and infer high quality appearances but cannot handle novel categories [157, 219]. Differentiable rendering [115, 124, 150] systems seek to implement the rendering process (rasterization and shading) in a differentiable manner so that gradients can be propagated to and from neural networks. Such methods can be used to directly optimize parameters such as pose or appearance. Current differentiable rendering methods are limited by the difficulty of implemented complex appearance models and require a 3D mesh. We seek to combine the best of these methods by creating a differentiable rendering pipeline that does not require a 3D mesh by instead building voxelized latent representations from a small number of reference images.

Multi-View Reconstruction Our method takes inspiration from multi-view reconstruction methods. It is most similar to space carving [110] and can be seen as a latent-space extension of it. Dense fusion methods such as [148, 225] generate dense point clouds of

the objects from RGB-D sequences. Recent works [210, 211] have explored ways to learn object representations from unaligned views. These methods recover coarse geometry and pose given an image, but require large amounts of training data for a single object category. Our method builds on both approaches: we train a network to reconstruct an object; however, instead of training per-object or per-category, we provide multiple reference images at inference time to create a 3D latent representation which can be rendered from novel viewpoints.

6.2 Overview

We present an end-to-end system for novel view reconstruction and pose estimation. We present our system in two parts. Sec. 6.3 describes our reconstruction pipeline which takes a small collection of reference images as input and produces a flexible representation which can be rendered from novel viewpoints. We leverage multi-view consistency to construct a latent representation and do not rely on category specific shape priors. This key architecture decision enables generalization beyond the distribution of training objects. We show that our reconstruction pipeline can accurately reconstruct unseen object categories from real images. In Sec. 6.4, we formulate the 6D pose estimation problem using our neural renderer. Since our rendering process is fully differentiable, we directly optimize for the camera parameters without the need for additional training or code-book generation for new objects.

Camera Model Throughout this paper we use a perspective pinhole camera model with an intrinsic matrix

$$\mathbf{K} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{6.1}$$

and a homogeneous extrinsic matrix $\mathbf{E} = [\mathbf{R}|\mathbf{t}]$, where f_u and f_v are the focal lengths, u_0 and v_0 are the coordinates of the camera principal point, and $\mathbf{R}$ and $\mathbf{t}$ are rotation

and translation of the camera, respectively. We also define a *viewport* cropping parameter $\mathbf{c} = (u_-, v_-, u_+, v_+)$ which represents a bounding box around the object in pixel coordinates. For brevity, we refer to the collection of these camera parameters as $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{t}, \mathbf{c}\}$.

6.3 Neural Reconstruction and Rendering

Given a set of N reference images with associated object poses and object segmentation masks, we seek to construct a representation of the object which can be rendered with arbitrary camera parameters. Building on the success of recent methods [151, 190], we represent the object as a latent 3D voxel grid. This representation can be directly manipulated using standard 3D transformations–naturally accommodating our requirement of novel view rendering. The overview of our method is shown in Fig. 6.2. There are two main components in our reconstruction pipeline: 1) *Modeling* the object by predicting per-view feature volumes and fusing them into a single canonical latent representation; 2) *Rendering* the latent representation to depth and color images.

6.3.1 Modeling

Our modeling step is inspired by space carving [110] in that our network takes observations from multiple views and leverages multi-view consistency to build a canonical representation. However, instead of using photometric consistency, we use latent features to represent each view which allows our network to learn features useful for this task.

Per-View Features We begin by generating a feature volume for each input view $\boldsymbol{\mathcal{I}}_i \in \{\boldsymbol{\mathcal{I}}_1, \ldots, \boldsymbol{\mathcal{I}}_N\}$. Each feature volume corresponds to the camera frustum of the input camera, bounded by the viewport parameter $\mathbf{c} = (u_-, v_-, u_+, v_+)$ and depth-wise by $z \in [z_c - r, z_c + r]$ where z_c is the distance to the object center and r is the radius of the object. Figure 6.3 illustrates the generation of the per-view features. Similar to [188], we use U-Nets [174] for their property of preserving spatial structure. We first compute 2D features $g_{\text{pix}}(\mathbf{x}_i) \in \mathbb{R}^{C \times H \times W}$ by passing the input $\mathbf{x}_i$ (an RGB image $\boldsymbol{\mathcal{I}}_i$, a binary mask $\boldsymbol{\mathcal{M}}_i$, and optionally

depth $\boldsymbol{\mathcal{D}}_i$) through a 2D U-Net. The deprojection unit ($p_\uparrow$) then lifts the 2D image features in $\mathbb{R}^{C\times H\times W}$ to 3D volumetric features in $\mathbb{R}^{(C/D)\times D\times H\times W}$ by factoring the 2D channel dimension into the 3D channel dimension $C' = C/D$ and depth dimension D. This deprojection operation is the exact opposite of the projection unit presented in [150]. The lifted features are then passed through a 3D U-Net g_{cam} to produce the volumetric features for the camera:

$$\boldsymbol{\Phi}_\mathbf{i} = g_{\text{cam}} \circ p_\uparrow \circ g_{\text{pix}}(\mathbf{x}_i) \in \mathbb{R}^{C'\times M\times M\times M}. \tag{6.2}$$

Camera to Object Coordinates Each voxel in our feature volume represents a point in 3D space. Following recent works [150, 151, 188], we transform our feature volumes directly using rigid transformations. Consider a continuous function $\phi(\mathbf{x}) \in \mathbb{R}^{C'}$ defining our camera-space latent representation, where $\mathbf{x} \in \mathbb{R}^3$ is a point in camera coordinates. The feature volume $\boldsymbol{\Phi}$ is a discrete sample of this function. This representation in object space is given by $\psi(\mathbf{x}') = \phi(\mathbf{W}^{-1}\mathbf{x}')$ where $\mathbf{x}'$ is a point in object coordinates and $\mathbf{W} = [\mathbf{R}|\mathbf{t}]$ is an object-to-camera extrinsic matrix. We compute the object-space volume $\hat{\boldsymbol{\Psi}}$ by sampling $\phi(\mathbf{W}^{-1}\mathbf{x}'_{ijk})$ for each object-space voxel coordinate $\mathbf{x}'_{ijk}$. In practice, this is done by trilinear sampling the voxel grid and edge-padding values that fall outside. Given this transformation operation $T_{\text{c}\to\text{o}}$, the object-space feature volume is given by $\hat{\boldsymbol{\Psi}}_\mathbf{i} = T_{\text{c}\to\text{o}}(\boldsymbol{\Phi}_\mathbf{i})$.

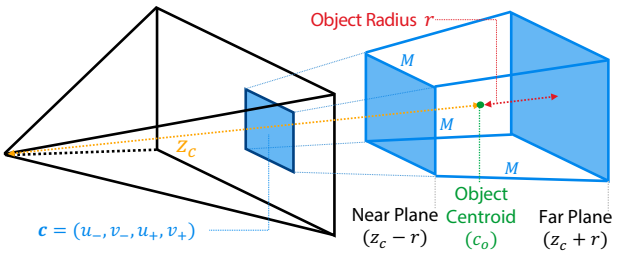


Figure 6.3: The $M \times M \times M$ per-view feature volumes computed in the modeling network corresponds a depth bounded camera frustum. The blue box on the image plane is determined by the camera crop parameter $\mathbf{c} = (u_-, v_-, u_+, v_+)$ and together with the depth determines the bounds of the frustum.

View Fusion We now have a collection of feature volumes $\hat{\boldsymbol{\Psi}}_\mathbf{i} \in \{\hat{\boldsymbol{\Psi}}_\mathbf{i}, \ldots, \hat{\boldsymbol{\Psi}}_\mathbf{N}\}$, each associated with an input view. Our fusion module f fuses all views into a single canonical feature volume: $\boldsymbol{\Psi} = f(\hat{\boldsymbol{\Psi}}_\mathbf{1}, \ldots, \hat{\boldsymbol{\Psi}}_\mathbf{N})$.

Simple channel-wise average pooling yields good results but we found that sequentially

integrating each volume using a Recurrent Neural Network (RNN) similarly to [188] slightly improved reconstruction accuracy (see §6.5.5). Using a recurrent unit allows the network to keep and ignore features from views in contrast to average pooling. This facilitates comparisons between different views allowing the network to perform operations similar to the photometric consistency criterion used in space carving [110]. We use a Convolutional Gated Recurrent Unit (ConvGRU) [7] so that the network can leverage spatial information.

6.3.2 Rendering

Our rendering module takes the fused object volume $\boldsymbol{\Psi}$ and renders it given arbitrary camera parameters $\boldsymbol{\theta}$. Ideally, the rendering module would directly regress a color image. However, it is challenging to preserve high frequency details through a neural network. U-Nets [174] introduce skip connections between equivalent-scale layers allowing high frequency spatial structure to propagate to the end of the network, but it is unclear how to add skip connections in the presence of 3D transformations. Existing works such as [126, 188] train a single network for each scene allowing the decoder to memorize high frequency information while the latent representation encodes state information. Trying to predict color without skip connections results in blurry outputs. We side-step this difficulty by first rendering depth and then using an image-based rendering approach to produce a color image.

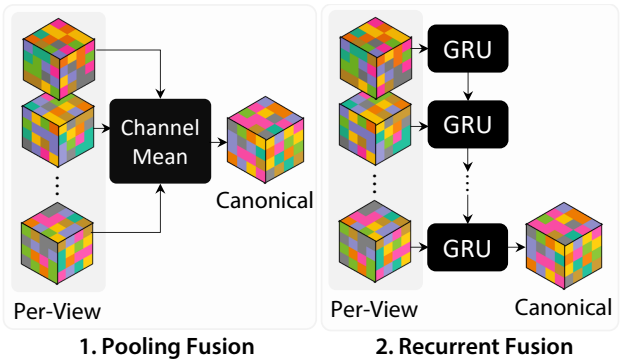


Figure 6.4: We illustrate two methods of fusion per-view feature volumes. (1) Simple channel-wise average pooling and (2) a recurrent fusion module similar to that of [188]

Decoding Depth Depth is a 3D representation, making it easier for the network to exploit the geometric structure we provide. In addition, depth tends to be locally smoother compared

to color allowing more information to be compactly represented in a single voxel.

Our rendering network is a simple inversion of the reconstruction network and bears many similarities to RenderNet [150]. First, we pass the canonical object-space volume $\mathbf{\Psi}$ through a small 3D U-Net (h_{obj}) before transforming it to camera coordinates using the method described in Sec. 6.3.1. We perform the transformation with an object-to-camera extrinsic matrix $\mathbf{E}$ instead of the inverse $\mathbf{E}^{-1}$. A second 3D U-Net (h_{cam}) then decodes the resulting volume to produce a feature volume:

$$\mathbf{\Psi'} = h_{\text{cam}} \circ T_{\text{o}\to\text{c}} \circ h_{\text{obj}}(\mathbf{\Psi}) \,, \tag{6.3}$$

which is then flattened to a 2D feature grid $\mathbf{\Phi'} = p_{\downarrow}(\mathbf{\Psi'})$ using the projection unit ($p_{\downarrow}$) from [150] by first collapsing the depth dimension into the channel dimension and applying a 1x1 convolution. The resulting features are decoded by a 2D U-Net (h_{pix}) with two output branches for depth (h_{depth}) and for a segmentation mask (h_{mask}). The outputs of the rendering network are given by

$$\mathbf{y}_{\text{depth}}(\mathbf{\Phi'}) = h_{\text{depth}} \circ h_{\text{pix}}(\mathbf{\Phi'}) \,, \tag{6.4}$$

$$\mathbf{y}_{\text{mask}}(\mathbf{\Phi'}) = h_{\text{mask}} \circ h_{\text{pix}}(\mathbf{\Phi'}) \,. \tag{6.5}$$

Image Based Rendering (IBR) We use image-based rendering [185] to leverage the reference images to predict output color. Given the camera intrinsics $\mathbf{K}$ and depth map for an output view, we can recover the 3D object-space position of each output pixel (u, v) as

$$\mathbf{X} = \mathbf{E}^{-1} \left(\frac{u - u_0}{f_u} z, \frac{v - v_0}{f_v} z, z, 1 \right)^T \,, \tag{6.6}$$

which can be transformed to the input image frame as $\mathbf{x'_i} = \mathbf{K_i W_i X}$ for each input camera $\boldsymbol{\theta}_i = \{\mathbf{K_i}, \mathbf{W_i}\}$. The output pixel can then copy the color of the corresponding input pixel to produce a reprojected color image.

The resulting reprojected image will contain invalid pixels due occlusions. There are multiple strategies to weighting each pixel including 1) weighting by reprojected depth error, 2) weighting by similarity between input and query cameras, 3) using a neural network. The

first choice suffers from artifacts in the presence of depth errors or thin surfaces. The second approach yields reasonable results but produces blurry images for intermediate views. We opt for the third option. Following deep blending [76], we train a network that predicts blend weights $\boldsymbol{\mathcal{W}}_i$ for each reprojected input $\boldsymbol{\mathcal{I}}'_i$: $\boldsymbol{\mathcal{I}}_o = \sum_i \boldsymbol{\mathcal{W}}_i \odot \boldsymbol{\mathcal{I}}'_i$, where $\odot$ is an element-wise product. The blend weights are predicted by a 2D U-Net. The inputs to this network are 1) the depth predicted by our reconstruction pipeline, 2) each reprojected input image $\boldsymbol{\mathcal{I}}'_i$, and 3) a view similarity score s based on the angle between the input and query poses.

6.3.3 Implementation Details

Training Data We train our reconstruction network on shapes from ShapeNet [30] which contains around 51,300 shapes. We exclude large models for efficient data loading resulting in around 30,000 models. We generate UV maps using Blender's smart UV projection [16] to facilitate texturing. We normalize all models to unit diameter. When rendering, we sample a random image from MS-COCO [120] for each component of the model. We render with the Beckmann model [11] with randomized parameters and also render uniformly colored objects with a probability of 0.5.

Network Input We generate our training data at a resolution of 640×480. However, the input to our network is a fixed size 128×128. To keep our inputs consistent and our network scale-invariant, we 'zoom' into the object such that all images appear to be from the same distance. This is done by computing a bounding box size

$$(w_b, h_b) = \left(\frac{d'w'}{f_u dw}, \frac{d'h'}{f_v dh} \right) , \tag{6.7}$$

where (w, h) is the current image width and height, d is the distance to the centroid c_o (See Fig. 6.3), (w', h') is the desired output size, and d' is the desired 'zoom' distance and cropping around object centroid projected to image coordinates (c_u, c_v). This defines the viewport parameter

$$\mathbf{c} = (c_u - w_b/2, c_v - h_b/2, c_u + w_b/2, c_v + h_b/2) . \tag{6.8}$$

The cropped image is scaled to 128×128.

Training In each iteration of training, we sample a 3D model and then sample 16 random reference poses and 16 random target poses. Each pose is sampled by uniformly sampling a unit quaternion and translation such that the object stays within frame. We train our network using the Adam optimizer [104] with a fixed learning rate of 0.001 for 1.5M iterations. Each batch consists of 20 objects with 16 input views and 16 target views. We use an L_1 reconstruction loss for depth and binary cross-entropy for the mask. We apply the losses to both the input and output views. We randomly orient our canonical coordinate frame in each iteration by uniformly sampling a random unit quaternion. This prevents our network from overfitting to the implementation of our latent voxel transformations. We also add motion blur, color jitter, and pixel noise to the color inputs and add noise to the input masks using the same procedure as [134].

6.4 Object Pose Estimation

Given an image $\boldsymbol{\mathcal{I}}$, and a depth map $\boldsymbol{\mathcal{D}}$, a pose estimation system provides a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ which together define an object-to-camera coordinate transformation $\mathbf{E} = [\mathbf{R}|\mathbf{t}]$ referred to as the *object pose*. In this section, we describe how we use our reconstruction pipeline described in Sec. 6.3 to directly optimize for the pose. We first find a *coarse* pose using only forward inference and then *refine* it using gradient optimization.

Formulation Pose is defined by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. Our formulation also includes the viewport parameter $\mathbf{c}$ defined in §6.2. Defining a viewport allows us to efficiently pass the input to the reconstruction network while also providing scale invariance. We encode the rotation as a quaternion $\mathbf{q}$ and translation as $\mathbf{t}$. We assume we are given an RGB image $\boldsymbol{\mathcal{I}}$, an object segmentation mask $\boldsymbol{\mathcal{M}}$, and depth $\boldsymbol{\mathcal{D}}$ comprising the input $\mathbf{x} = \{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{D}}\}$.

6.4.1 Losses

In order to estimate pose, we must provide a criterion which quantifies the quality of the pose. We use four loss functions. One is a standard L_1 depth reconstruction loss

$$\mathcal{L}_{\text{depth}}(\boldsymbol{\mathcal{D}}^*, \boldsymbol{\mathcal{D}}) = \|\boldsymbol{\mathcal{D}}^* - \boldsymbol{\mathcal{D}}\|_1 \,, \tag{6.9}$$

which disambiguates the object scale and measures how well the predicted depth $\boldsymbol{\mathcal{D}}$ matches the input depth $\boldsymbol{\mathcal{D}}^*$. We also use a pixel-wise binary cross entropy loss $\mathcal{L}_{\text{mask}}$ on the predicted mask as well as intersection over union (IoU) loss

$$\mathcal{L}_{\text{iou}}(\boldsymbol{\mathcal{M}}^*, \boldsymbol{\mathcal{M}}) = \log U - \log I \,, \tag{6.10}$$

where U is the sum of the pixels in the union and I is the sum of the pixels in the intersection of the masks $\boldsymbol{\mathcal{M}}^*$ and $\boldsymbol{\mathcal{M}}$. Finally, we introduce a novel latent loss which leverages our reconstruction network F. Given the input $\mathbf{x} = \{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{D}}\}$, a latent object $\boldsymbol{\Psi}$, and a pose $\boldsymbol{\theta}$, the latent loss is defined as

$$\mathcal{L}_{\text{latent}}(\mathbf{x}, \boldsymbol{\theta}; \boldsymbol{\Psi}) = \left\| H_{\boldsymbol{\theta}}(G_{\boldsymbol{\theta}}(\mathbf{x})) - H_{\boldsymbol{\theta}}(\boldsymbol{\Psi}) \right\|_1 \,, \tag{6.11}$$

where $H_{\boldsymbol{\theta}}$ is the rendering network up to the projection layer and $G_{\boldsymbol{\theta}}$ is the modeling network as described in Sec. 6.3. This loss differs from auto-encoder based approaches such as [50, 199] in that 1) our network is not trained on the object, and 2) the loss is computed directly given the image and camera pose. Our pose estimation problem is given by:

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{depth}} + \lambda \mathcal{L}_{\text{latent}} + \gamma \mathcal{L}_{\text{mask}} + \eta \mathcal{L}_{\text{iou}}, \tag{6.12}$$

where λ, γ, η are the weights of the lossees. The parameters of the losses are omitted for clarity.

Parameterization We parameterize the rotation in the log quaternion form $\boldsymbol{\omega} = (0, \omega_1, \omega_2, \omega_3)$ which ensures that all updates to the parameters result in a valid unit quaternion:

$$\mathbf{q} = \exp(\boldsymbol{\omega}) = \begin{pmatrix} \cos\|\boldsymbol{\omega}\| \\ \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \sin\|\boldsymbol{\omega}\| \end{pmatrix}. \tag{6.13}$$

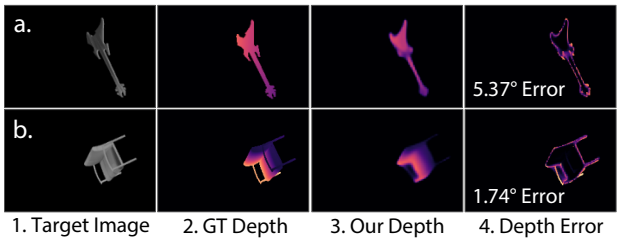


Figure 6.5: Two examples from the ModelNet experiment. (1) target image, (2) ground truth depth, (3) optimized predicted depth, and (4) L_1 error between the ground truth and our prediction. (a) illustrates how a pose with low depth error can still result in a relatively high angular error.

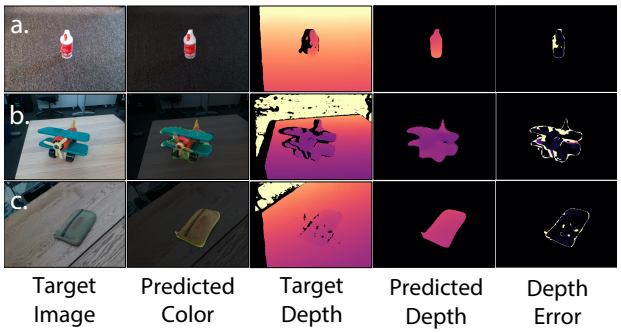


Figure 6.6: Qualitative results on the MOPED dataset.

Coarse Initialization Although we have a differentiable renderer, the space of poses is non-convex which can lead to bad local minima when using gradient based optimization. We therefore bootstrap the pose by computing a coarse estimate. This also has the benefit of speeding up inference since it only requires forward evaluation.

We begin by estimating the translation of the object $\mathbf{t} = (x, y, z)$ as the centroid of the bounding cube defined by the mask bounding box $\mathbf{c}$ and corresponding depth values. We initialize k poses using the estimated translation. To get good coverage of possible orientations, we evenly sample azimuth and elevation angles using a Fibonacci lattice [68] then uniformly sample a random yaw angle. We use the cross entropy method [45] to optimize the translation and log quaternion parameters, and use a Gaussian Mixture Model as the probability distribution.

Pose Optimization Our entire pipeline is differentiable end-to-end. We can therefore optimize Eq. (6.12) using gradient optimization. Given a latent object $\mathbf{\Psi}$ and a coarse pose estimate $\boldsymbol{\theta}$, we compute the loss and propagate the gradients back to the camera pose. This step only requires the rendering network and does not use the modeling network. The image-based rendering network is also not used in this step. We jointly optimize the rotation $\mathbf{q}$, translation $\mathbf{t}$, and viewport $\mathbf{c}$ using Adam [104].

Table 6.1: Evaluation on LINEMOD. We report the ADD recall metric [82], comparing against DeepIM and pix2pose. Symmetric objects are indicated by * and the pose is considered correct if it is flipped along the z axis.

Method	Input	ape	benchvice	camera	can	cat	driller	duck	eggbox*	glue*	holepunch	iron	lamp	phone	mean
Ours	RGB-D	**88.0**	92.4	74.4	88.8	**94.5**	91.7	68.1	96.3	94.9	**82.1**	74.6	94.7	**91.5**	87.1
DeepIM [117]	RGB	77.0	**97.5**	**93.5**	**96.5**	82.1	**95.0**	**77.7**	**97.1**	**99.4**	52.8	**98.3**	**97.5**	87.7	**88.6**
pix2pose [161]	RGB	58.1	91.0	60.9	84.4	65.0	76.3	43.8	96.8	79.4	74.8	83.4	82.0	45.0	72.4

6.5 Experiments

We evaluate our method on LINEMOD [81], ModelNet [227] and our new dataset MOPED. We aim to evaluate pose estimation accuracy on unseen objects.

6.5.1 Evaluation Metrics

We use four main evaluation metrics. 1) $(k^\circ, k\text{cm})$: a pose is considered correct if it is within k° and $k$cm of the ground truth target pose where the angular metric is the angle between the orientations. 2) ADD [82]: the average distance between points after being transformed by the ground truth and predicted poses. 3) ADD-S: A modification to ADD that computes the average distance to the closest point rather than the ground truth point to account for symmetric objects. 4) Proj.2D: the pixel distance between the projected points of the ground truth and predicted pose.

6.5.2 Experiments on the LINEMOD Dataset

We evaluate our method on the LINEMOD dataset. We compare our results with DeepIM [117] and pix2pose [161]. Both of these methods are trained on the LINEMOD dataset. DeepIM uses the 3D models in an online fashion to refine the pose. We *do not train* on the dataset. Instead, our network is given 16 reference views of each object during inference time. We use the provided segmentation masks during inference. We follow the train/test split of [84] and sample our reference views from the training set. We follow the same evaluation methodol-

ogy of [82], reporting the percentage of poses with ADD metric less than 10% of the object diameter. Table. 6.1 shows the results. Our experiments show that our method performs on-par with state of the art supervised methods despite having never seen the objects.

6.5.3 Experiments on the ModelNet Dataset

We conduct experiments on ModelNet [227] to evaluate the generalization of our method toward unseen object categories. To this end, we train our network on all the meshes in ShapeNetCore [30] excluding the categories we are going to evaluate on. We closely follow the evaluation protocol of [117] here. The model is evaluated on 7 unseen categories: *bathtub, bookshelf, guitar, range hood, sofa, wardrobe*, and *TV stand*. For each category, 50 pairs of initial and target object pose are sampled. We compared with [117] and [200] where all the methods initialized with the initial pose and evaluated on how successful they are on estimating the target pose. We report three metrics: $(5^\circ, 5cm)$, *ADD* within 10% of the object diameter, and *Proj.2D* within 5 pixels.

Table 6.2 shows the quantitative results on the ModelNet dataset. On average, our method achieves state-of-the-art results on all the metrics thanks to our ability to perform continuous optimization on pose. However, for the $(5^\circ, 5cm)$ metric, there are object categories that our method performs worse despite performing well on all other metrics. One reason is the image and spatial resolution. The input and output images to our network have resolution 128×128. The resolution of our voxel representation is $16 \times 16 \times 16$. The limited resolution can hinder the performance for small objects or objects that are distant from the camera. Small changes in the depth of each pixel may disproportionately affect the rotation of the object compared to our losses. Fig. 6.5 shows examples from the ModelNet experiment illustrating this limitation.

6.5.4 Experiments on the MOPED Dataset

Table 6.2: ModelNet pose refinement experiments compared to DeepIM (DI) [117] and Multi-Path Learning (MP) [200].

	(5°, 5cm)			ADD (0.1d)			Proj2D (5px)		
	DI	MP	Ours	DI	MP	Ours	DI	MP	Ours
bathtub	71.6	**85.5**	85.0	88.6	91.5	**92.7**	73.4	80.6	**94.9**
bookshelf	39.2	**81.9**	80.2	76.4	85.1	**91.5**	51.3	76.3	**91.8**
guitar	50.4	69.2	**73.5**	69.6	80.5	**83.9**	77.1	80.1	**96.9**
range_hood	69.8	**91.0**	82.9	89.6	95.0	**97.9**	70.6	83.9	**91.7**
sofa	82.7	**91.3**	89.9	89.5	95.8	**99.7**	94.2	86.5	**97.6**
tv_stand	73.6	85.9	**88.6**	92.1	90.9	**97.4**	76.6	82.5	**96.0**
wardrobe	62.7	88.7	**91.7**	79.4	92.1	**97.0**	70.0	81.1	**94.2**
Mean	64.3	84.8	**85.5**	83.6	90.1	**94.3**	73.3	81.6	**94.7**

Table 6.3: AUC metrics on MOPED by reference view count.

# Views	1	2	4	8	16	32
ADD	23.9	34.2	49.8	**63.1**	62.7	61.1
ADD-S	78.7	82.8	89.2	**91.1**	90.8	90.8
Proj.2D	9.5	15.9	30.4	44.7	**46.8**	42.6

Table 6.4: AUC metrics for different view fusion strategies

	ADD	ADD-S	Proj.2D
Avg Pool	57.5	90.1	40.1
ConvGRU	**63.1**	**91.1**	**44.7**

Figure 6.7: Objects in MOPED–a new dataset for model-free pose estimation. The objects shown are: (a) toy_plane, (b) duplo_dude, (c) cheezit, (d) duster, (e) black_drill, (f) orange_drill, (g) graphics_card, (h) remote, (i) rinse_aid, (j) vim_mug, and (k) pouch.

We introduce the Model-free Object Pose Estimation Dataset (MOPED). MOPED consists of 11 objects, shown in Fig. 6.7. For each object, we take multiple RGB-D videos cover all views the object. We first use KinectFusion [148] to register frames from a single capture and then use a combination of manual annotation and automatic registration [175, 243, 244] to align separate captures. We generate object segmentation maps using [232]. For each object, we select reference frames with farthest point sampling to ensure good coverage of the object. For test sequences, we capture each object in 5 different environments. We sample every other frame for evaluation videos. This results in approximately 300 test images per object. We evaluate our method and baselines using three metrics for which we provide the Area Under Curve (AUC): 1) *ADD* with threshold between $0-10cm$, 2) *ADD-S* with threshold between $0-10cm$, and 3) *Proj.2D* with threshold between $0-40px$. We compute all metrics for all sampled frames.

We compare our method with PoseRBPF [50], a state-of-the-art model-based pose estimation method. Since PoseRBPF requires textured 3D models, we reconstruct a mesh for each object by aggregating point clouds from reference captures and building a TSDF volume. The point clouds are integrated into the volume using KinectFusion [148]. The meshes have artifacts such as washed out high frequency details and shadow vertices due to slight misalignment (see supplementary materials). Table 6.5 shows quantitative comparisons on the MOPED dataset. Note that our method is not trained on the test objects while PoseRBPF has a separate encoder for each object. Our method achieves superior performance on both ADD and ADD-S. We evaluate different version of our method with different combinations of loss functions. Compared to our combined loss, optimizing only $\mathcal{L}_{\text{depth}}$ performs better for geometrically asymmetric objects but worse on textured objects

such as the cheezit box. Optimizing both losses achieves better results on textured objects. Fig. 6.6 shows estimated poses for different test images. Please see supplementary materials for qualitative examples.

6.5.5 Ablation Studies

In this section, we analyze the effect of different design choices and how they affect the robustness of our method.

Number of reference views We first evaluate the sensitivity of our method to the number of input reference views. Novel view synthesis is easier with more reference views because there is a higher chance that a query view will be close to a reference view. Table 6.3 shows that the accuracy increases with the number of reference views. In addition, having more than 8 reference views only yields marginal performance gains shows that our method does not require many views to achieve good pose estimation results.

View Fusion We compare multiple strategies for aggregating the latent representations from each reference view. The naive way is to use a simple pooling function such as average/max pooling. Alternatively, we can integrate the volumes using an RNN such as a ConvGRU so that the network can reason across views. Table 6.4 shows the quantitative evaluation of these two variations. Although the average performance of the objects are very similar, the ConvGRU variation performs better than the average pooling variation. This indicates the importance of spatial relationship in the voxel representation for pose estimation.

6.6 Conclusion

We have presented a novel framework for building 3D object representations at inference time using a small number of reference images, as well as an accompanying neural renderer to render the 3D representation from arbitrary 6D viewpoints. Our networks are trained

on thousands of shapes with random textures rendered under various lighting conditions allowing it to robustly generalize to unseen objects *without additional training*.

We leverage our reconstruction and rendering pipeline for zero-shot pose estimation. We perform pose estimation given just a small number of reference views and without needing to train any network. This greatly simplifies the process for performing pose estimation on novel objects as a detailed 3D model is not required. In addition, we have a single universal network which works for all objects including unseen ones. For future work, we plan to investigate unseen object pose estimation in cluttered scenes with occlusions. We also plan to speed up the pose estimation process by applying network optimization techniques.

Table 6.5: Quantitative Results on MOPED Dataset. We report the Area Under Curve (AUC) for each metric. † shows the average with the symmetric objects rinse_aid and cheezit excluded. Ours (D) is our method with the depth loss only and Ours (D+L) is our method with both the depth and latent losses. In some cases, the latent loss helps resolve pose ambiguities not possible with only depth.

	PoseRBPF			Ours (D)			Ours (D+L)		
Input	Textured 3D Mesh			**Image + Camera Pose**					
Training	Yes			**No**					
# Networks	Per-Object			**Single Universal**					
Object	ADD	ADD-S	Proj.2D	ADD	ADD-S	Proj.2D	ADD	ADD-S	Proj.2D
black_drill	75.3	91.7	54.2	**90.3**	**95.4**	**78.1**	89.4	95.1	76.2
duplo_dude	84.1	93.5	61.5	**89.4**	**94.9**	**77.4**	88.9	94.7	76.2
duster	**72.2**	**90.6**	**51.5**	40.0	88.3	17.8	47.2	87.9	15.5
graphics_card	71.8	**81.3**	51.2	**73.1**	79.3	62.2	73.0	79.2	**62.5**
orange_drill	70.3	87.5	43.9	**78.4**	**93.7**	**62.3**	78.3	93.5	61.7
pouch	26.5	80.7	14.1	**61.9**	**91.8**	**44.2**	54.8	91.7	31.4
remote	49.6	82.1	25.1	58.2	**92.0**	19.0	**60.0**	91.9	**26.3**
toy_plane	48.6	88.6	29.5	55.5	89.7	46.5	**82.9**	**92.2**	**70.5**
vim_mug	**59.0**	**92.9**	**39.4**	51.5	91.8	22.7	40.0	91.7	8.0
rinse_aid	87.2	**94.7**	**68.7**	**71.3**	93.2	41.7	67.4	93.3	34.7
cheezit	**64.3**	**93.8**	**55.6**	24.8	92.4	20.5	24.6	92.2	20.2
Mean	**64.4**	88.9	**45.0**	63.1	91.1	44.7	64.2	**91.2**	43.9
Mean w/o sym	61.9	87.7	41.2	66.5	90.8	47.8	**68.3**	**90.9**	**47.6**

Chapter 7
CONCLUSION

In this thesis, I proposed several methods and applications for *scene rerendering* which allow scenes to be photorealistically rendered from novel viewpoints and with novel appearances. I investigated *scene level* methods useful in applications such as photography, as well as *object level* methods which can aid in content creation or augmented and virtual reality. Below I summarize each of my contributions:

- **Casual Capture of Deformable Scenes**: In Chapter 3, I introduced *Nerfies*, a method that enables the capture of deformable scenes by casually waving a camera. I showed that an as-rigid-as-possible deformation prior and a coarse-to-fine deformation regularization were keys to obtaining high-quality results. I then showcased the application of casual selfie captures which enables high-fidelity reconstructions of human subjects using a cellphone capture.

- **Casual Capture of Topologically Varying Scenes**: One of the limitations of Nerfies is that it cannot handle scenes which exhibit topological variations such as the opening and closing of a person's mouth. In Chapter 4, I presented a solution to this limitation called *HyperNeRF*. HyperNeRF models topological variations as slices through a higher-dimensional space. To keep the space compact and avoid overfitting, HyperNeRF delays the use of these ambient dimensions, and uses deformable hyperplanes to extract these slices. HyperNeRF combines insights from level-set methods and deformation-based dynamic NeRF models to reconstruct both large motions and topological variations.

- **Photorealistic Relightable Models**: There is an abundance of 3D models available

through various online repositories. However, these 3D shapes often lack *relightable textures* which are required to render them realistically. Relightable models are crucial for use in video games, e-commerce, and augmented/virtual reality. It is notoriously difficult to create relightable textures, and the process usually involves many manual, time-consuming steps. In Chapter 5, I introduced *PhotoShape*, a technique to automatically assign high-quality relightable textures to a collection of 3D models with limited material information. The textures come from an extensive database of materials, and the material-to-3D assignments are performed with the guidance of real images to ensure plausible material configurations. This approach yields thousands of high-quality, photorealistic 3D models.

- **Neural Rendering for Zero-Shot Pose Estimation**: Nerfies (introduced in Chapter 3) and HyperNeRF (introduced in Chapter 4) are examples of *neural rendering*. Neural rendering pipelines are usually differentiable, meaning that we can optimize their input parameters with respect to their outputs. In Chapter 6, I presented *LatentFusion*, a method that exploits this property of a similar neural rendering pipeline to optimize the pose of an object. LatentFusion uses neural networks for both reconstruction and rendering, and thus we can directly optimize the pose of an input image. We use this fact to perform pose estimation on *unseen objects*. While existing methods require training separate neural networks for each novel object, LatentFusion can achieve the same results without training and with a single universal network.

7.1 Future Work

The ultimate goal of *scene rerendering* is to create representations that allow users to modify and render any aspect of a scene. Methods such as Nerfies and HyperNeRF enable us to change some elements, such as perspective, framing, and subject pose. However, there are still many elements that we have not covered, such as exposure and focus. We must also work to improve the speed and robustness of these methods to better facilitate their use in

practice.

7.1.1 Exposure and HDR

Current NeRF models, including Nerfies and HyperNeRF, predict radiance directly in sRGB space, assuming that the exposure of all photographs in the input collection are identical. This design also provides no mechanism to modify the exposure of the output. We can relax this assumption by explicitly predicting linear color values and modeling exposure duration. Each input image i may be assigned an exposure duration parameter t_i, which can then be optimized as a free parameter like the deformation latent codes of Nerfies. We can then convert the output to sRGB space to evaluate the loss. This approach confers multiple advantages. First, we now have direct control over the exposure duration and can modify the exposure during rendering. Second, the resulting model predicts linear, high-dynamic-range (HDR) values that can be rendered with any tone-mapping function or displayed on an HDR display. Some of these ideas have started being explored, such as in Mildenhall et al. [144] and Huang et al. [88].

7.1.2 Focus

Current NeRF models assume a pinhole camera model, meaning rays emanate from the focal point in straight lines through each pixel. A pinhole camera model cannot model defocus blur because it is caused by multiple light rays from different points of the scene arriving at the same pixel. To model focus, we must do two things: (1) introduce a camera lens model that transforms ray directions according to the properties of the lens; (2) each pixel must consider multiple sampled rays to account for light coming from multiple directions.

Jeong et al. [95] models and optimizes non-linear distortions caused by optical aberrations. Such an approach could be expanded to model the behavior of a camera lens more completely. The main challenge likely comes from the second requirement, as sampling multiple rays per pixel increases the computation cost linearly in the number of samples required. Solutions

to the sampling problem may come from using importance sampling techniques or clever use of the representation akin to mip-NeRF [9].

7.1.3 Robustness

Nerfies and HyperNeRF require multiple views to constrain the 3D structure of a scene. This may cause issues in captures that do not cover enough viewpoints. Some extensions to NeRF, such as NSFF [114] and Video-NeRF [230], leverage supervision in the form of depth or optical flow fields to regularize the training of the model. This significantly improves the robustness of the model when the capture baseline is small or in the presence of large movements. Insights from these approaches may be combined with Nerfies and HyperNeRF to create a more robust model.

7.1.4 Speed and Practicality

Fast inference for NeRF has already been addressed by numerous recent works [61, 77, 169, 238], some of which enable real-time rendering on a laptop computer. A more difficult task is improving the training time of NeRF-based methods. Many recent methods have achieved faster training by using more traditional data structures either exclusively [197] or in conjunction with a coordinate-based MLP [146]. Significant gains in training time will likely also come from a more efficient sampling of the underlying scene. Currently, most samples along a ray correspond to empty space. DONeRF [147] predicts a 2D depth map to inform sampling during inference; it may be possible to leverage a similar approach to avoid empty spacing during training as well. In addition, the integrated positional encoding of Mip-NeRF could be leveraged to perform optimization in a coarse-to-fine fashion to speed up convergence. Better initializing the model can also speed up convergence significantly, such as through meta-learning [203]. Perhaps we may seek to avoid training altogether by learning generic models that do not require per-scene optimization [217].

Beyond thinking about how we can train NeRF faster, it is also essential to consider domain-specific solutions. For example, if our task is to make minor adjustments to photos,

it may be possible to simplify the problem and save on data collection and computation. In addition, we may leverage additional data that we have access to for free: most modern mobile phones have multiple lenses, and some have dedicated depth sensors such as LiDAR. We could leverage extra information from these sensors to speed up training and make the system more robust.

BIBLIOGRAPHY

[1] Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of . . . , 1991.

[2] Adobe Stock. Adobe stock: Stock photos, royalty-free images, graphics, vectors and videos. https://stock.adobe.com/, 2018.

[3] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Practical svbrdf capture in the frequency domain. *ACM Trans. Graph.*, 32(4), 2013.

[4] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Trans. Graph.*, 34(4), 2015.

[5] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv:1906.08240*, 2019.

[6] Sameer Ansari, Neal Wadhwa, Rahul Garg, and Jiawen Chen. Wireless software synchronization of multiple distributed cameras. *ICCP*, 2019.

[7] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[8] Jonathan T. Barron. A general and adaptive robust loss function. *CVPR*, 2019.

[9] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv*, 2021.

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[11] Petr Beckmann and Andre Spizzichino. The scattering of electromagnetic waves from rough surfaces. *Norwood, MA, Artech House, Inc.*, 1987.

[12] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics*, 32(4), 2013.

[13] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. In *CVPR*, 2015.

[14] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020.

[15] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. *SIGGRAPH*, 1999.

[16] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2019. URL `http://www.blender.org`.

[17] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. *ICML*, 2018.

[18] James Booth, Anastasios Roussos, Allan Ponniah, David Dunaway, and Stefanos Zafeiriou. Large scale 3D morphable models. *IJCV*, 2018.

[19] Sofien Bouaziz, Yangang Wang, and Mark Pauly. Online modeling for realtime facial animation. *ACM TOG*, 2013.

[20] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM TOG*, 2014.

[21] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. *arXiv preprint arXiv:2006.13240*, 2020.

[22] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[23] Adam Brady, Jason Lawrence, Pieter Peers, and Westley Weimer. genbrdf: discovering new analytic brdfs with genetic programming. *ACM Transactions on Graphics*, 33(4), 2014.

[24] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. *CVPR*, 2000.

[25] brusspup. Amazing t-rex illusion!, Dec 2013.

[26] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001.

[27] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney.

[28] Thomas J Cashman and Andrew W Fitzgibbon. What shape are dolphins? building 3D morphable models from 2D images. *TPAMI*, 2012.

[29] Manmohan Chandraker. On shape and material recovery from motion. In *ECCV*, Cham, 2014. Springer International Publishing.

[30] 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. Technical Report arXiv:1512.03012 [cs.GR], 2015.

[31] 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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[32] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph.*, 2010.

[33] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997.

[34] 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), 2015.

[35] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[36] Shenchang Eric Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 29–38, 1995.

[37] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[38] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019.

[39] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5939–5948, 2019.

[40] Christopher Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 628–644, 2016.

[41] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[42] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015.

[43] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM ToG*, 2015.

[44] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[45] Pieter-Tjerk De Boer, Dirk Kroese, Shie Mannor, and Reuven Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

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

[47] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996.

[48] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. URL `https://github.com/google-research/google-research/tree/master/jaxnerf`.

[49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[50] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A rao-blackwellized particle filter for 6D object pose tracking. In *Robotics: Science and Systems (RSS)*, 2019.

[51] Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *ACM Transactions on Graphics*, 34(2), 2015.

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

[53] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4D: Real-time performance capture of challenging scenes. *ACM ToG*, 2016.

[54] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[55] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[56] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9), 2010.

[57] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5515–5524, 2016.

[58] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *ACM TOG*, 2019.

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

[60] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *CVPR*, pages 8649–8658, June 2021.

[61] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

[62] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[63] Stuart Geman and Donald E. McClure. Bayesian image analysis: An application to single photon emission tomography. *Proceedings of the American Statistical Association*, 1985.

[64] Todor Georgiev, Zhan Yu, Andrew Lumsdaine, and Sergio Goma. Lytro camera technology: theory, algorithms, performance analysis. In *Multimedia content and mobile devices*, volume 8667, page 86671J. International Society for Optics and Photonics, 2013.

[65] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Efstratios Gavves, Mario Fritz, Luc Van Gool, and Tinne Tuytelaars. Reflectance and natural illumination from single-material specular objects using deep learning. *IEEE TPAMI*, 2017.

[66] Stamatios Georgoulis, Vincent Vanweddingen, Marc Proesmans, and Luc Van Gool. Material classification under natural illumination using reflectance maps. In *WACV*, 2017.

[67] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[68] Álvaro González. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49, 2010.

[69] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 27, 2014.

[70] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[71] Richard Langton Gregory. The intelligent eye. 1970.

[72] 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 ToG*, 2019.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[74] Peter Hedman and Johannes Kopf. Instant 3d photography. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[75] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *ACM Transactions on Graphics (TOG)*, 36(6):1–15, 2017.

[76] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[77] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv*, 2021.

[78] Herman-Miller. Herman miller 3d models. `https://www.hermanmiller.com/resources/models/3d-` 2018.

[79] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Progressive encoding for neural optimization, 2021.

[80] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *SIGGRAPH*, 2001.

[81] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *International Conference on Computer Vision (ICCV)*, pages 858–865, 2011.

[82] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562, 2012.

[83] Heiko Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2386–2393. IEEE, 2006.

[84] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Federico Tombari Fabian Manhardt, Tae-Kyun Kim, Jiri Matas, and Carsten Rother. BOP: Benchmark for 6D object pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[85] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6D object pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3385–3394, 2019.

[86] Hui Huang, Ke Xie, Lin Ma, Dani Lischinski, Minglun Gong, Xin Tong, and Daniel Cohen-Or. Appearance modeling via proxy-to-image alignment. *ACM Trans. Graph.*, 37(1), 2018.

[87] Qixing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Trans. Graph.*, 34(4), 2015.

[88] Xin Huang, Qi Zhang, Feng Ying, Hongdong Li, Xuan Wang, and Qing Wang. Hdr-nerf: High dynamic range neural radiance fields. *arXiv preprint arXiv:2111.14451*, 2021.

[89] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[90] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[91] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *CVPR*, 2017.

[92] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

[93] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Material memex: Automatic material suggestions for 3d objects. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2012)*, 31(5), 2012.

[94] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*, pages 3121–3128. IEEE, 2011.

[95] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, pages 5846–5854, 2021.

[96] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, Leonidas Guibas, et al. Shapeflow: Learnable deformations among 3d shapes. *arXiv preprint arXiv:2006.07982*, 2020.

[97] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[98] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6): 1–10, 2016.

[99] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[100] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[101] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Trans. Graph.*, 33(4), 2014.

[102] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt. Deep video portraits. *ACM ToG*, 2018.

[103] Kihwan Kim, Jinwei Gu, Stephen Tyree, Pavlo Molchanov, Matthias Nießner, and Jan Kautz. A lightweight approach for on-the-fly reflectance estimation. In *ICCV*, 2017.

[104] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[106] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph.*, 26 (3), 2007.

[107] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3d photography. *ACM Transactions on Graphics (TOG)*, 39(4):76–1, 2020.

[108] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*. 2011.

[109] Abhijit Kundu, Yin Li, and James M Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3559–3568, 2018.

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

[111] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph

cuts. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

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

[113] Hao Li, Linjie Luo, Daniel Vlasic, Pieter Peers, Jovan Popović, Mark Pauly, and Szymon Rusinkiewicz. Temporally coherent completion of dynamic shapes. *ACM TOG*, 2012.

[114] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis, 2021.

[115] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37 (6):1–11, 2018.

[116] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Trans. Graph.*, 36(4), 2017.

[117] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[118] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[119] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[120] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[121] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE TPAMI*, 33(5), 2011.

[122] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *ICCV*, 2017.

[123] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[124] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *International Conference on Computer Vision (ICCV)*, 2019.

[125] Stephen Lombardi and Ko Nishino. Reflectance and illumination recovery in the wild. *IEEE TPAMI*, 2015.

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

[127] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM ToG*, 2019.

[128] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graph.*, 2015.

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

[130] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[131] BD LUCAS. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop, 1981*, 1981.

[132] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020.

[133] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[134] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017.

[135] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. LookinGood: Enhancing performance capture with real-time neural re-rendering. *SIGGRAPH Asia*, 2018.

[136] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *CVPR*, 2021.

[137] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3), 2003.

[138] Philip F McLauchlan. Gauge invariance in projective 3d reconstruction. In *Proceedings IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes (MVIEW'99)*, pages 37–44. IEEE, 1999.

[139] Abhimitra Meka, Christian Haene, Rohit Pandey, Michael Zollhoefer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. Deep reflectance fields - high-quality facial reflectance field inference from color gradient illumination. 2019.

[140] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.

[141] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019.

[142] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[143] 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. *ECCV*, 2020.

[144] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. *arXiv preprint arXiv:2111.13679*, 2021.

[145] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D bounding box estimation using deep learning and geometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[146] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[147] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Computer Graphics Forum*, 40(4):45–59, 2021. doi: https://doi.org/10.1111/cgf.14340. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14340`.

[148] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[149] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. *CVPR*, 2015.

[150] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[151] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. In *International Conference on Computer Vision (ICCV)*, 2019.

[152] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. *ICCV*, 2019.

[153] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[154] Geoffrey Oxholm and Ko Nishino. Shape and reflectance estimation in the wild. *IEEE TPAMI*, 2016.

[155] Andrea Palazzi, Luca Bergamini, Simone Calderara, and Rita Cucchiara. End-to-end 6-DOF object pose estimation through differentiable rasterization. In *European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

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

[157] 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.

[158] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. LatentFusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *CVPR*, 2020.

[159] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[160] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[161] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *International Conference on Computer Vision (ICCV)*, pages 7668–7677, 2019.

[162] Despoina Paschalidou, Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2018.

[163] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *ICPR*, volume 3, pages 314–317. IEEE, 2000.

[164] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4561–4570, 2019.

[165] Poliigon. A library of textures, materials and hdr's for artists that want photorealism. https://www.poliigon.com/, 2018.

[166] Andrew Prock and Chuck Dyer. Towards real-time voxel coloring. In *Proceedings of the DARPA Image Understanding Workshop*, volume 1, page 2, 1998.

[167] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[168] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[169] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *arXiv preprint arXiv:2103.13744*, 2021.

[170] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. Deep reflectance maps. In *CVPR*, 2016.

[171] Konstantinos Rematas, Chuong Nguyen, Tobias Ritschel, Mario Fritz, and Tinne Tuytelaars. Novel views of objects from a single image. *TPAMI*, 2017.

[172] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*. Springer, 2016.

[173] Olinde Rodrigues. De l'attraction des sphéroïdes. In *Correspondence Sur l'École Impériale Polytechnique*, pages 361–385, 1816.

[174] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241, 2015.

[175] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[176] Nikolay Savinov, Christian Hane, Lubor Ladicky, and Marc Pollefeys. Semantic 3d reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5460–5469, 2016.

[177] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Dart: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, 2015.

[178] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *CVPR*, 2016.

[179] Gabriel Schwartz and Ko Nishino. Automatically discovering local visual material attributes. *CVPR*, 2015.

[180] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. *NeurIPS*, 2020.

[181] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[182] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Trans. Graph.*, 36(4), 2017.

[183] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward H. Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 108(3), 2013.

[184] Lavanya Sharan, Ruth Rosenholtz, and Edward H. Adelson. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 14(10), 2014.

[185] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing*, volume 4067, pages 2–13, 2000.

[186] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. *ACM SIGGRAPH 2012 Courses*, 2012.

[187] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. BigBIRD: A large-scale 3D database of object instances. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516, 2014.

[188] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3D feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[189] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3D feature embeddings. In *CVPR*, 2019.

[190] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NuerIPS)*, 2019.

[191] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019.

[192] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

[193] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006.

[194] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. *EUROGRAPHICS*, 2007.

[195] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*, 2015.

[196] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM TOG*, 2007.

[197] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *arXiv preprint arXiv:2111.11215*, 2021.

[198] 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. *SIGGRAPH*, 2019.

[199] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D orientation learning for 6D object detection from rgb images. In *European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[200] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, and Rudolph Triebel. Multi-path learning for object pose estimation across domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[201] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. http://theia-sfm.org.

[202] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[203] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021.

[204] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the art on neural rendering. *Computer Graphics Forum*, 2020.

[205] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-time face capture and reenactment of rgb videos. *CVPR*, 2016.

[206] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG*, 2019.

[207] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *TPAMI*, 2008.

[208] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018. URL `https://arxiv.org/abs/1809.10790`.

[209] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video, 2021.

[210] Shubham Tulsiani, Tinghui Zhou, Alexei Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[211] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2897–2905, 2018.

[212] Ali Osman Ulusoy, Andreas Geiger, and Michael J Black. Towards probabilistic volumetric reconstruction using ray potentials. In *2015 International Conference on 3D Vision*, pages 10–18. IEEE, 2015.

[213] Bruno Vallet and Bruno Lévy. What you seam is what you get. Technical report, INRIA - ALICE Project Team, 2009.

[214] Vray-materials.de. Vray-materials.de - your ultimate v-ray material resource. https://www.vray-materials.de, 2018.

[215] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, 2007.

[216] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352, 2019.

[217] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[218] Ting-Chun Wang, Jun-Yan Zhu, Ebi Hiroaki, Manmohan Chandraker, Alexei Efros, and Ravi Ramamoorthi. A 4D light-field dataset and CNN architectures for material recognition. In *ECCV*, 2016.

[219] Tuanfeng Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy Mitra. Unsupervised texture transfer from images to model collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–13, 2016.

[220] 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. ISSN 0730-0301.

[221] Yunhai Wang, Minglun Gong, Tianhua Wang, Daniel Cohen-Or, Hao Zhang, and Baoquan Chen. Projective analysis for 3d shape segmentation. *ACM Trans. Graph.*, 32(6), 2013.

[222] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. *Asilomar Conference on Signals, Systems & Computers*, 2003.

[223] Mitra NJ Wang TY, Ritschel T. Joint material and illumination estimation from photo sets in the wild. *Eurographics*, 2017.

[224] Tomas Werner, Roger D Hersch, and Vaclav Hlavac. Rendering real-world objects using view interpolation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 957–962. IEEE, 1995.

[225] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015.

[226] Kyle Wilson, David Bindel, and Noah Snavely. When is rotations averaging hard? In *European Conference on Computer Vision*, pages 255–270. Springer, 2016.

[227] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[228] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020.

[229] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020.

[230] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[231] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[232] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. The best of both modes: Separately leveraging RGB and depth for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2019.

[233] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. GHUM & GHUML: Generative 3D human shape and articulated pose models. *CVPR*, 2020.

[234] Jia Xue, Hang Zhang, Kristin J. Dana, and Ko Nishino. Differential angular imaging for material recognition. *CVPR*, 2017.

[235] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, pages 4541–4550, 2019.

[236] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint arXiv:2003.09852*, 2020.

[237] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, 2020.

[238] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.

[239] zbyg. Hdri pack. https://zbyg.deviantart.com/art/HDRi-Pack-2-103458406, 2018.

[240] Hang Zhang, Kristin Dana, and Ko Nishino. Reflectance hashing for material recognition. In *CVPR*, 2015.

[241] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[242] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[243] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision (ECCV)*, pages 766–782, 2016.

[244] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[245] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[246] Zhiming Zhou, Guojun Chen, Yue Dong, David Wipf, Yong Yu, John Snyder, and Xin Tong. Sparse-as-possible svbrdf acquisition. *ACM Trans. Graph.*, 35(6), 2016.

[247] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Trans. Graph.*, 2014.

[248] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. State of the art on monocular 3D face reconstruction, tracking, and applications. *Computer Graphics Forum*, 2018.

[249] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3D menagerie: Modeling the 3D shape and pose of animals. *CVPR*, 2017.