Scene Representations for View Synthesis with Deep Learning

Pratul Srinivasan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-214
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-214.html

December 17, 2020

Scene Representations for View Synthesis with Deep Learning

by

Pratul Srinivasan

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

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Ren Ng, Chair
Professor Ravi Ramamoorthi
Professor Alexei Efros
Professor Martin S. Banks

Fall 2020

Scene Representations for View Synthesis with Deep Learning

Abstract

Scene Representations for View Synthesis with Deep Learning

by

Pratul Srinivasan

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ren Ng, Chair

In this dissertation, we investigate the question of how 3D scenes should be represented, such that the representation can be effectively estimated from standard photographs and can then be used to synthesize images of the same scene from novel unobserved viewpoints. Recovering photorealistic scene representations from images has been a longstanding goal of computer vision and graphics, and has typically been addressed using representations from standard computer graphics pipelines, such as triangle meshes, which are not particularly amenable to end-to-end optimization for maximizing the fidelity of rendered images. Instead, we advocate for the use of scene representations that are specifically well-suited to being used in differentiable deep learning pipelines. We explore the efficacy of various representations for view synthesis tasks including synthesizing local views around a single input image, extrapolating views around a pair of nearby input images, and interpolating novel views from a set of unstructured images. We present scene representations that succeed at the aforementioned tasks, which share two common properties: they represent scenes as volumes and that they avoid the poor scaling properties of regularly-sampled voxel grids by using compressed or parameter-efficient volume representations.

_____

Professor Ren Ng
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

I am incredibly fortunate to have been advised by Ravi Ramamoorthi and Ren Ng. Both of them gave me the freedom to develop my taste in research and work on problems that I found interesting, while providing the guidance and questioning to help me learn from my missteps. Ravi took me on as a graduate student without any experience in computer graphics, and supported me and my ideas even after he moved from Berkeley to San Diego. Ren took me on as a student when he joined after my first year at Berkeley, and has helped me grow as a researcher in countless ways. I especially appreciate Ren's continual effort to teach me about the importance of clarity in thought, presentation, and communication.

I feel lucky to have been a part of the amazing research group that Ren created here at Berkeley. Spending time with this group of people has made my graduate school experience immeasurably more fun and joyful. I want to thank Ben Mildenhall, Grace Kuo, Cecilia Zhang, Utkarsh Singhal, Matt Tancik, and Vivien Nguyen for being such great labmates and friends. I've greatly enjoyed collaborating with Matt Tancik and Ben Mildenhall over the past few years, and I feel like I have learned so much from both of them. Ben and I started working on the problems of light field reconstruction and view synthesis around the same time, and I cannot imagine a better research collaborator.

I would also like to thank my other dissertation committee members, Marty Banks and Alyosha Efros. Marty's curiosity inspires me, and I always learn incredible facts about the human visual system every time I talk to him. Alyosha's work at the intersection of computer vision and graphics has been a great source of inspiration.

I also would like to thank the other research mentors I've had throughout the years. Sina Farsiu took me into his lab as an undergraduate at Duke University, and spent countless hours teaching me about the basics of image processing. Working with Sina led me to the field of computer vision, and I am grateful for the effort he took to help me start my career as a researcher. Similarly, Michael Tao, who was one of Ravi's senior students at Berkeley when I joined, went out of his way to involve me in his projects. He taught me about light fields and light field cameras when I just started grad school, and working with him on the applications of light field cameras sparked my interest in this research field.

I am also grateful for the mentorship and support provided by my longtime collaborators at Google Research: Jon Barron, Noah Snavely, and Richard Tucker. They have all been incredibly supportive of my ideas, and I really enjoyed the sum-

mers I spent working in Mountain View with Jon and New York with Noah and Richard.

I would also like to thank the good friends I have made at Berkeley outside the lab: Efthymios Philip Papageorgiou, Andy Michaels, and Alyssa Zhou, for making my time at Berkeley so enjoyable.

I am grateful to my family for their endless love and support. My parents, Srinivasan Rajagopalan and Preeti Shrikhande, have always encouraged me to pursue my interests. My brother, Aditya Srinivasan, has been an incredible friend and never ceases to amaze me with his growth and accomplishments. Finally, my fiancée Hani Ahir has been a tremendous source of love, inspiration, and support throughout my time at Berkeley.

Chapter 1

Introduction

Visual storytelling, from art to video games, plays a central part in our lives. While computer graphics techniques have enabled us to use photorealistic 3D content to immerse viewers into a visually compelling story, creating these realistic 3D graphics assets typically requires painstaking and expensive work by talented artists. A promising strategy for democratizing 3D content creation is to develop algorithms that use observed images to recover a 3D representation with the same functionalities as hand-designed graphics assets. This task, commonly referred to as "inverse rendering", is a longstanding problem in the fields of computer vision and graphics and a successful solution would enable anyone to use real-world objects and scenes in photography, filmmaking, and game development.

In this dissertation, we focus on the specific inverse rendering problem of using observed images to recover a scene representation that is able to render novel photorealistic views of a scene, a task typically called "view synthesis" or "image-based rendering" (IBR). In particular, we investigate different strategies for representing 3D scenes, with the goal of illuminating the important characteristics shared by effective scene representations.

Although view synthesis does not capture the full functionality of traditional graphics assets (a representation for view synthesis does not support relighting or physical simulation), we argue that view synthesis is a core problem in visual computing with high potential impact. Progress towards effective scene representations for view synthesis will be useful for the fully general inverse rendering task, and a

Figure 1.1. A visual sample of outputs and results of the work presented in the following chapters. This dissertation focuses on the problem of novel view synthesis: using a set of captured input images of a scene to render views of the scene from novel unobserved viewpoints.

successful solution to view synthesis is crucial for capturing real-world scenes for virtual and augmented reality (VR and AR).

While the problem of view synthesis has a rich history in computer vision and graphics, its progress has been limited by prior work's adhering to 3D geometry representations that are not amenable to being optimized for the goal of view synthesis. In scenarios where the input images are not captured densely enough for simple interpolation-based light field rendering algorithms, the dominant paradigm for view synthesis algorithms was to use the best available off-the-shelf method to estimate scene geometry from a set of input images, and then construct an algorithm that synthesizes novel views using this provided scene geometry. For example, a common pipeline for view synthesis was to run Structure-from-Motion (SfM) on the set of input images to estimate camera parameters and a sparse point cloud, pass these results to a Multi-View Stereo (MVS) algorithm to estimate dense depth maps for each input image, fuse these depth maps into a triangle mesh using surface reconstruction and meshing algorithms, and finally render novel views by reprojecting and blending observed images using the reconstructed mesh. The key issue with this paradigm is that the scene geometry is not estimated with the specific goal of maximizing the quality of rendered novel views, and each step in this pipeline optimizes a slightly different set of criteria. The resulting scene geometry contains inaccuracies (ex. edges that are not quite aligned with image edges) that are particularly problematic for rendering novel views.

We argue that we should strive to use scene representations with geometry that is specifically optimized to maximize the quality of synthesized novel views. This can be achieved by following the differentiable programming paradigm popularized by deep learning: if we construct the scene representation estimation and rendering procedures out of differentiable functions, this would enable us to use gradient-based optimization to tune the program end-to-end to minimize the error of rendering example novel views.

Unfortunately, the polygonal mesh scene representations traditionally used for view synthesis are not amenable to this differentiable programming paradigm. It is unclear how to use a differentiable parametric function approximator like a deep neural network to accept observed images as inputs and predict a polygon mesh, and the procedures for rendering images of mesh representations from novel viewpoints contain discontinuities that prevent us from easily optimizing a mesh to maximize the fidelity of rendered images. Therefore, it is vital for us to rethink the representations we use for view synthesis and investigate what characteristics are necessary for representations to be effectively used with differentiable deep learning pipelines.

Below, we provide an high-level overview of prior approaches to view synthesis, followed by an overview of this dissertation and our contributions towards designing scene representations for view synthesis that work effectively within deep learning pipelines. Figure 1.1 visualizes example results from the projects presented in this dissertation.

1.1 Scene Representations for View Synthesis

Our work builds upon decades of progress in view synthesis, and Shum and Kang [121] provide an excellent review of classic view synthesis algorithms. As discussed in their review, it is useful to characterize view synthesis algorithms by the extent to which they make use of explicit scene geometry. Existing view synthesis algorithms can be roughly binned into three categories: methods that use a global fixed 3D representation, methods that use view-dependent 3D representations where either the geometry or appearance changes based on viewpoint, and light field methods that avoid explicit geometry estimation.

1.1.1 Global Scene Representations

A straightforward approach for view synthesis is to use the observed images to estimate a global 3D model, similar to standard graphics assets, so novel views can be rendered by simply projecting this fixed 3D representation into any viewpoint. Various scene representations have been used in this context, including voxel grids [115], height fields [13, 87], and texture-mapped triangle meshes [144]. While using a single fixed 3D representation is a natural approach, estimating accurate geometry can be quite difficult in practice. This motivated the field of view synthesis to investigate view-dependent 3D representations, which can still render high-fidelity novel views with inaccurate geometry.

1.1.2 View-Dependent Scene Representations

Many modern algorithms for view synthesis with sparsely-sampled images are based on the framework of using approximate scene geometry to reproject and blend a set of nearby sampled views to render an image from the target viewpoint. This strategy was used in foundational work on view-dependent texture-mapping [21] and unstructured lumigraph rendering [7]. Even if the estimated geometry is incorrect, reprojecting nearby source images into a target viewpoint can convincingly render occlusion and non-Lambertian effects if the source viewpoints are close enough to the target viewpoint. However, it is very difficult to estimate high-quality meshes whose geometric boundaries align well with edges in images, and these methods typically suffer from significant artifacts when rendering target viewpoints further away from the observed input views. State-of-the-art algorithms [47, 48] attempt to remedy this shortcoming with complicated pipelines that involve both global mesh and local depth map estimation, but they are still limited by the difficulties inherent to estimating mesh geometry from images.

Other algorithms that use view-dependent scene representations [11, 13, 68, 87, 97, 103] avoid difficult and expensive global mesh estimation. Instead, they compute detailed local geometry for each input image (typically as single or multi layer depth maps) and render novel views by reprojecting and blending nearby input images. This strategy can be quite effective, and the work presented in Chapter 4 discusses how to utilize this strategy within a deep learning pipeline.

1.1.3 Light Field Rendering

Methods based on light field rendering [76] eschew any geometric reasoning, and simply sample images on a regular grid so that new views can be rendered as slices of the sampled light field. Light field rendering can be thought of as a view-dependent 3D representation where the proxy geometry is simply a plane. Due to the simple proxy geometry, light field rendering techniques require sampling dense input views, which can quickly become intractable for applications such as virtual reality where we would like to support large viewer motions.

1.1.4 Deep Learning for View Synthesis

Prior to the work presented in this dissertation, the research community was just beginning to explore leveraging deep learning for the task of novel view synthesis. Initial attempts [137, 167] focused on synthesizing far-away views of individual objects from a single input view, and eschewed any geometry reasoning. More recent approaches improved the quality of synthesized views by learning to model local geometry from the target viewpoint and using this geometry to reproject and blend input views. This includes a light field view synthesis algorithm [61] trained to interpolate between four input corner views sampled on a plane, and the DeepStereo algorithm trained to interpolate views along the path of captured Google Street View images [31]. A key theme in this dissertation is that using persistent volumetric scene representations within a deep learning pipeline can substantially improve upon the results of these prior works, which suffered from artifacts caused by their separately predicting geometry for each target novel view.

1.2 Dissertation Overview and Contributions

The main contribution of this dissertation is an exploration of different scene representations for view synthesis, with the goal of illuminating the important characteristics that make scene representations effective for deep learning based view synthesis. Figure 1.2 visualizes the general problem formulation we consider, where the input is a set of images with corresponding camera poses, and our goal is to recover a scene representation that supports rendering novel views of the scene.

In Chapter 2, we ask the question "Do we need a global 3D scene representation for effective view synthesis with deep learning?". We investigate whether

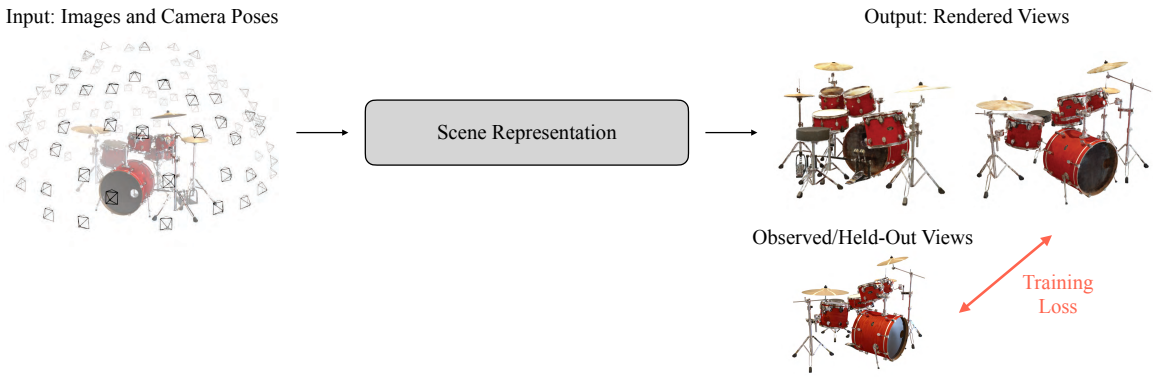


Figure 1.2. We investigate scene representations within the following view synthesis problem formulation: Given input images of a scene along with the corresponding camera viewpoints, our goal is to recover a scene representation that is able to render novel unobserved views of the scene. We advocate for optimizing the scene representation (or a function that predicts the scene representation) by minimizing the loss of using that scene representation to render novel views.

we can just represent a scene as a light field, a function that simply describes the distribution of light rays in the scene. Representing a scene as a light field is equivalent to representing a scene as the total distribution of images of that scene. We explore this idea by training a deep network to map input images of a scene to output images at novel viewpoints without reconstructing a single persistent model of 3D scene geometry. Our experiments show that this straightforward treatment of view synthesis as a learned mapping from input images to output images can produce convincing results for a limited range of novel viewpoints. However, we show that this approach is not adequate for rendering larger amounts of viewer motion, because the lack of a persistent 3D representation results in perceptually-jarring "flickering" artifacts when adjacent rendered views are synthesized using different predicted geometry. It becomes increasingly difficult to encourage the separately-predicted geometry from each target viewpoint to be consistent with a single persistent 3D representation.

To remedy this deficiency, we need to use a persistent 3D scene representation. By persistent, we mean that the same scene geometry should be used to render a range of novel viewpoints. Next, we attempt to answer the question "What scene representations should we be using?". There is no single answer to this question; many 3D representations are used in computer graphics for different applications, each with their own strengths and weaknesses. However, we would like to deter-

mine the characteristics that make a scene representation successful when used in a deep learning pipeline for view synthesis.

Chapters 3, 4, and 5 investigate the specific choice of using sampled volumetric representations for view synthesis. Volumetric representations are particularly attractive for view synthesis with deep learning for two reasons. First, grid-sampled volumes naturally fit within the array-programming paradigms of modern deep learning frameworks and can easily be used with modern convolutional network architectures. Second, volumetric representations are much better-suited for gradient-based optimization than surface representations such as meshes or signed distance fields. Intuitively, a volumetric representation can arbitrarily add or remove opacity from anywhere within the volume to best match an image, but a surface representation has to gradually move the surface from its initial location, which is prone to local minima. However, the optimization benefits of volumetric representations come at the expense of increased storage costs. Therefore, it is important to represent these volumetric representations in a parameter-efficient manner. One effective parameter-efficient strategy for representing volumes is to use variable-resolution sampling and simply allocate more samples where finer resolution is needed for rendering novel views. In Chapters 3 and 4, we discuss a specific technique to represent large-scale scenes with this strategy for the task of synthesizing "forwards-facing" novel views of the scene. In Chapter 5, we explore a slightly different variable-resolution volumetric representation for estimating the illumination incident at any location within a scene (this task can be thought of as panoramic view synthesis since we are trying to synthesize a panorama representing the incoming light at any location within the scene). These three chapters provide strong evidence for the efficacy of parameter-efficient sampled volumetric representations for novel view synthesis tasks.

In Chapter 6, we explore a radically different strategy for representing volumetric scenes in a parameter-efficient manner. Instead of using a discrete sampled volume, we represent the scene as a continuous volumetric function, parameterized by a fully-connected neural network that takes in a 3D coordinate and 2D viewing direction, and outputs the volume density and view-dependent color at that location. Thus, the entire scene is encoded in the weights of this deep network. We demonstrate that this can be much more efficient than a sampled volumetric representation while still enabling us to render photorealistic novel views of the scene.

Finally, in chapter 7, we summarize the lessons we have learned through our exploration of scene representations for deep learning based view synthesis, and discuss a vision for the future of 3D scene representations for view synthesis and inverse rendering in general.

Chapter 2

Local View Extrapolation with Light Fields

In this chapter, we investigate whether a fixed 3D scene representation is needed for effective view synthesis. Instead, we ask whether we can tackle the problem of novel view synthesis by simply training a deep network to map observed input views to output views. This overall strategy can be thought of as using the light field as the scene representation for novel view synthesis. Figure 2.2 illustrates how the task of synthesizing novel camera views is equivalent to synthesizing novel 2D slices of the 4D light field from observed 2D slices.

2.1 Introduction

We focus on a problem that we call "local light field synthesis", which we define as the promotion of a single photograph to a plenoptic camera light field. One can think of this as expansion from a single view to a dense 2D patch of views. We argue that local light field synthesis is a core visual computing problem with high

This chapter is based on joint work published at ICCV 2017 [133].

potential impact. First, it would bring light field benefits such as synthetic apertures and refocusing to everyday photography. Furthermore, local light field synthesis would systematically lower the sampling rate of photographs needed to capture large baseline light fields, by "filling the gap" between discrete viewpoints. This is a path towards making light field capture for virtual and augmented reality (VR and AR) practical. In this work, we hope to convince the community that local light field synthesis is actually a tractable problem.

From an alternative perspective, the light field synthesis task can be used as an unsupervised learning framework for estimating scene geometry from a single image. Without any ground-truth geometry for training, we can learn to estimate the geometry that minimizes the difference between the light field rendered with that geometry and the ground-truth light field.

Light field synthesis is a severely ill-posed problem, since the goal is to reconstruct a 4D light field given just a single image, which can be interpreted as a 2D slice of the 4D light field. To alleviate this, we use a machine learning approach that is able to utilize prior knowledge of natural light fields. In this work, we focus on scenes of flowers and plants, because they contain interesting and complex occlusions as well as a wide range of relative depths. Our specific contributions are the introduction of the largest available light field dataset, the prediction of 4D ray depths with a novel depth consistency regularization to improve unsupervised depth estimation, and a learning framework to synthesize a light field from a single image.

Light Field Dataset We collect the largest available light field dataset (Sec. 2.4), contaning 3343 light fields of flowers and plants, taken with the Lytro Illum camera. Our dataset limits us to synthesizing light fields with camera-scale baselines, but we note that our model can generalize to light fields of any scene and baseline given the appropriate datasets.

Ray Depths and Regularization Current view synthesis methods generate each view separately. Instead, we propose to concurrently predict the entire 4D light field by estimating a separate depth map for each viewpoint, which is equivalent to estimating a depth for each ray in the 4D light field (Sec. 2.5). We introduce a novel physically-based regularization that encourages the predicted depth maps to be consistent across viewpoints, alleviating typical problems that arise in depths created by view synthesis (Fig. 2.5). We demonstrate that our algorithm can predict depths from a single image that are comparable or better than depths estimated by

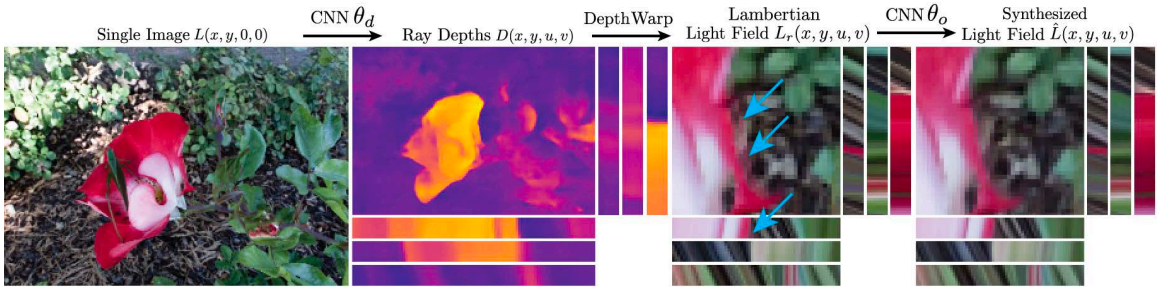


Figure 2.1. We propose a CNN framework that factors the light field synthesis problem into estimating depths for each ray in the light field, rendering a Lambertian approximation to the light field, and refining this approximation by predicting occluded rays and non-Lambertian effects (incorrect rays that are refined, in this case red rays that should be the color of the background instead of the flower, are marked with blue arrows). We train this network end-to-end by minimizing the reconstruction errors of the Lambertian and predicted light fields, along with a novel physically-based depth regularization. We demonstrate that we can predict convincing 4D light fields and ray depths from a single 2D image. We visualize synthesized light fields as a predicted corner view along with epipolar slices in both the u and v directions of different spatial segments.

a state-of-the-art physically-based non-learning method that uses the entire light field [58] (Fig. 2.6).

CNN Framework We create and study an end-to-end convolutional neural network (CNN) framework, visualized in Fig. 2.1, that factorizes the light field synthesis problem into the subproblems of estimating scene depths for every ray (Fig. 2.6, Sec. 2.5) (we use depth and disparity interchangeably, since they are closely related in structured light fields), rendering a Lambertian light field (Sec. 2.6.1), and predicting occluded rays and non-Lambertian effects (Sec. 2.6.2). This makes the learning process more tractable and allows us to estimate scene depths, even though our network is trained without any access to the ground truth depths. Finally, we demonstrate that it is possible to synthesize high-quality ray depths and light fields of flowers and plants from a single image (Fig. 2.1, Fig. 2.6, Fig. 2.9, Fig. 2.10, Sec. 2.7).

2.2 Related Work

Light Fields The 4D light field [82] is the total spatio-angular distribution of light rays passing through a region of free space. Previous work has demonstrated exciting applications of light fields, including rendering images from new viewpoints [76], changing the focus and depth-of-field of photographs after capture [93], correcting lens aberrations [92], and estimating scene flow [129].

View Synthesis from Light Fields Early work on light field rendering [76] captures a densely-sampled 4D light field of a scene, and renders images from new viewpoints as 2D slices of the light field. Closely related work on the Lumigraph [42] uses approximate geometry information to refine the rendered slices. The unstructured Lumigraph rendering framework [7] extends these approaches to use a set of unstructured (not axis-aligned in the angular dimensions) 2D slices of the light field. In contrast to these pioneering works which capture many 2D slices of the light field to render new views, we propose to synthesize a dense sampling of new views from just a single slice of the light field.

View Synthesis without Geometry Estimation Alternative approaches synthesize images from new viewpoints without explicitly estimating geometry. The work of Shi *et al*. [119] uses the observation that light fields are sparse in the continuous Fourier domain to reconstruct a full light field from a carefully-constructed 2D collection of views. Didyk *et al*. [23] and Zhang *et al*. [164] reconstruct 4D light fields from pairs of 2D slices using phase-based approaches.

Recent works have trained CNNs to synthesize slices of the light field that have dramatically different viewpoints than the input slices. Tatarchenko *et al*. [137] and Yang *et al*. [158] train CNNs to regress from a single input 2D view to another 2D view, given the desired camera rotation. The exciting work of Zhou *et al*. [167] predicts a flow field that rearranges pixels from the input views to synthesize novel views that are sharper than directly regressing to pixel values. These methods are trained on synthetic images rendered from large databases of 3D models of objects such as cars and chairs [10], while we train on real light fields. Additionally, they are not able to explicitly take advantage of geometry because they attempt to synthesize views at arbitrary rotations with potentially no shared geometry between the input and target views. We instead focus on the problem of synthesizing a dense sampling of views around the input view, so we can explicitly estimate geometry to produce higher quality results.

View Synthesis by Geometry Estimation Other methods perform view interpolation by first estimating geometry from input 2D slices of the light field, and then warping the input views to reconstruct new views. These include view interpolation algorithms [11, 41] which use wider baseline unstructured stereo pairs to estimate geometry using multi-view stereo algorithms.

More recently, CNN-based view synthesis methods been proposed, starting with the inspiring DeepStereo method that uses unstructured images from Google's Street View [31] to synthesize new views. This idea has been extended to view interpolation for light fields given 4 corner views [61], and the prediction of one image from a stereo pair given the other image [35, 40, 154].

We take inspiration from the geometry-based view synthesis algorithms discussed above, and also predict geometry to warp an input view to novel views. However, unlike previous methods, we synthesize an entire 4D light field from just a single image. Furthermore, we synthesize all views and corresponding depths at once, as opposed to the typical strategy of predicting a single 2D view at a time, and leverage this to produce better depth estimations.

3D Representation Inference from a Single Image Instead of synthesizing new imagery, many excellent works address the general inverse rendering problem of inferring the scene properties that produce an observed 2D image. The influential algorithm of Barron and Malik [4] solves an optimization problem with priors on reflectance, shape, and illumination to infer these from a single image. Other interesting works [25, 112] focus on inferring just the 3D structure of the scene, and train on ground-truth geometry captured with 3D scanners or the Microsoft Kinect. A number of exciting works extend this idea to infer a 3D voxel [15, 39, 153] or point set [28] representation from a synthetic 2D image by training CNNs on large databases of 3D CAD models. Finally, recent methods [111, 142, 156] learn to infer 3D voxel grids from a 2D image without any 3D supervision by using a rendering or projection layer within the network and minimizing the error of the rendered view. Our work is closely related to unsupervised 3D representation learning methods, but we represent geometry as 4D ray depths instead of voxels, and train on real light fields instead of views from synthetic 3D models of single objects.

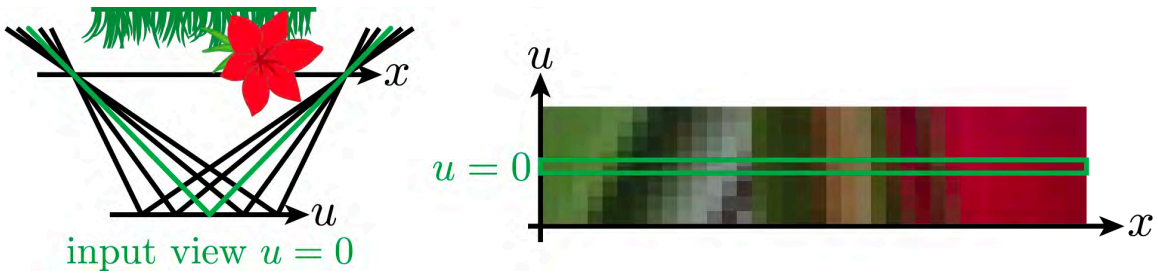


Figure 2.2. Two equivalent interpretations of the local light field synthesis problem. Left: Given an input image of a scene, with the field-of-view marked in green, our goal is to synthesize a dense grid of surrounding views, with field-of-views marked in black. The u dimension represents the center-of-projection of each virtual viewpoint, and the x axis represents the optical conjugate of the sensor plane. Right: Given an input image, which is a 1D slice of the 2D flatland light field (2D slice of the full 4D light field), our goal is to synthesize the entire light field. In our light field parameterization, vertical lines correspond to points in focus, and lines at a slope of 45 degrees correspond to points at the farthest distance that is within the depth of field of each sub-aperture image.

2.3 Light Field Synthesis

Given an image from a single viewpoint, our goal is to synthesize views from a densely-sampled grid around the input view. This is equivalent to synthesizing a 4D light field, given a central 2D slice of the light field, and both of these interpretations are visualized in Fig. 2.2. We do this by learning to approximate a function f:

$$\hat{L}(\mathbf{x}, \mathbf{u}) = f(L(\mathbf{x}, \mathbf{0})) \tag{2.1}$$

where $\hat{L}$ is the predicted light field, $\mathbf{x}$ is spatial coordinate (x, y), $\mathbf{u}$ is angular coordinate (u, v), and $L(\mathbf{x}, \mathbf{u})$ is the ground-truth light field, with input central view $L(\mathbf{x}, \mathbf{0})$.

Light field synthesis is severely ill-posed, but certain redundancies in the light field as well as prior knowledge of scene statistics enable us to infer other slices of the light field from just a single 2D slice. Figure 2.2 illustrates that scene points at a specific depth lie along lines with corresponding slopes in the light field. Furthermore, the colors along these lines are constant for Lambertian reflectance, and only change due to occlusions or non-Lambertian reflectance effects.

We factorize the problem of light field synthesis into the subproblems of esti-

mating the depth at each coordinate $(\mathbf{x}, \mathbf{u})$ in the light field, rendering a Lambertian approximation of the light field using the input image and these estimated depths, and finally predicting occluded rays and non-Lambertian effects. This amounts to factorizing the function f in Eq. 2.1 into a composition of 3 functions: d to estimate ray depths, r to render the approximate light field from the depths and central 2D slice, and o to predict occluded rays and non-Lambertian effects from the approximate light field and predicted depths:

$$\begin{aligned} D(\mathbf{x}, \mathbf{u}) =& d(L(\mathbf{x}, \mathbf{0})) \\ L_r(\mathbf{x}, \mathbf{u}) =& r(L(\mathbf{x}, \mathbf{0}), D(\mathbf{x}, \mathbf{u})) \\ \hat{L}(\mathbf{x}, \mathbf{u}) =& o(L_r(\mathbf{x}, \mathbf{u}), D(\mathbf{x}, \mathbf{u})) \end{aligned} \tag{2.2}$$

where $D(\mathbf{x}, \mathbf{u})$ represents predicted ray depths, and L_r represents the rendered Lambertian approximate light field. This factorization lets the network learn to estimate scene depths from a single image in an unsupervised manner.

The rendering function r (Sec. 2.6.1) is physically-based, while the depth estimation function d (Sec. 2.5) and occlusion prediction function o (Sec. 2.6.2) are both structured as CNNs, due to their state-of-the-art performance across many function approximation problems in computer vision. The CNN parameters are learned end-to-end by minimizing the sum of the reconstruction error of the Lambertian approximate light field, the reconstruction error of the predicted light field, and regularization losses for the predicted depths, for all training tuples:

$$\begin{aligned} \min_{\theta_d, \theta_o} \sum_{\mathcal{S}} \big[&||L_r - L||_1 + ||\hat{L} - L||_1 \\ &+ \lambda_c \psi_c(D) + \lambda_{tv} \psi_{tv}(D) \big] \end{aligned} \tag{2.3}$$

where θ_d and θ_o are the parameters for the depth estimation and occlusion prediction networks. ψ_c and ψ_{tv} are consistency and total variation regularization losses for the predicted ray depths, discussed below in Sec. 2.5. $\mathcal{S}$ is the set of all training tuples, each consisting of an input central view $L(\mathbf{x}, \mathbf{0})$ and ground truth light field $L(\mathbf{x}, \mathbf{u})$.

We include the reconstruction errors for both the Lambertian light field and the predicted light field in our loss to prevent the occlusion prediction network from attempting to learn the full light field prediction function by itself, which would prevent the depth estimation network from properly learning a depth estimation function.

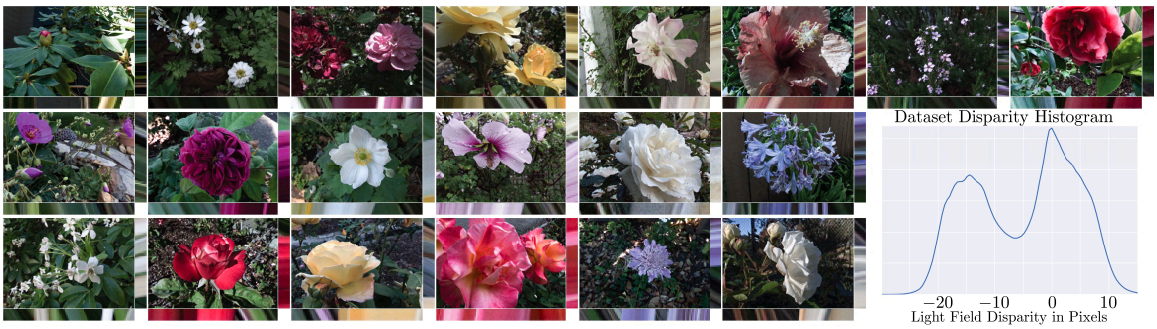


Figure 2.3. We introduce the largest available light field dataset, containing 3343 light fields of scenes of flowers and plants captured with the Lytro Illum camera in various locations and lighting settings. These light fields contain complex occlusions and wide ranges of relative depths, as visualized in the example epipolar slices. No ground truth depths are available, so we use our algorithm to predict a histogram of disparities in the dataset to demonstrate the rich depth complexity in our dataset. We will make this dataset available upon publication.

2.4 Light Field Dataset

To train our model, we collected 3343 light fields of flowers and plants with the Lytro Illum camera, randomly split into 3243 for training and 100 for testing. We captured all light fields using a focal length of 30 mm and $f/2$ aperture. Other camera parameters including the shutter speed, ISO, and white balance were set automatically by the camera. We decoded the sensor data from the Illum camera using the Lytro Power Tools Beta decoder, which demosaics the color sensor pattern and calibrates the lenslet locations. Each light field has 376x541 spatial samples, and 14x14 angular samples. Many of the corner angular samples lie outside the camera's aperture, so we used an 8x8 grid of angular samples in our experiments, corresponding to the angular samples that lie fully within the aperture.

This dataset includes light fields of several varieties of roses, poppies, thistles, orchids, lillies, irises, and other plants, all of which contain complex occlusions. Furthermore, these light fields were captured in various locations and times of day with different natural lighting conditions. Figure 2.3 illustrates the diversity of our dataset, and the geometric complexity in our dataset can be visualized in the epipolar slices. To quantify the geometric diversity of our dataset, we compute a histogram of the disparities across the full aperture using our trained depth estimation network, since we do not have ground truth depths. The left peak of this

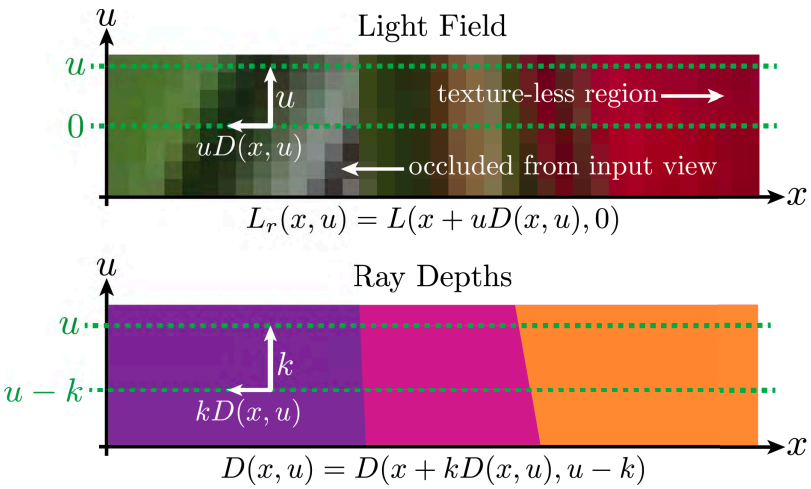


Figure 2.4. Top: In a Lambertian approximation of the light field, the color of a scene point is constant along the line corresponding to its depth. Given estimated disparities $D(x, u)$ and a central view $L(x, 0)$, we can render the flatland light field as $L(x, u) = L(x + uD(x, u), 0)$ ($D(x, u)$ is negative in this example). In white, we illustrate two prominent problems that arise when estimating depth by minimizing the reconstruction error of novel views. It is difficult to estimate the correct depth for points occluded from the input view, because warping the input view using the correct depth does not properly reconstruct the novel views. Additionally, it is difficult to estimate the correct depth in texture-less regions, because many possible depths result in the same synthesized novel views. Bottom: Analogous to the Lambertian color consistency, rays from the same scene point should have the same depth. This can be represented as $D(x, u) = D(x + kD(x, u), u - k)$ for any continuous value of k. We visualize ray depths using a colormap where darker colors correspond to further objects.

histogram corresponds to background points, which have large negative disparities, and the right peak of the histogram corresponds to the photograph subjects (typically flowers) which are in focus and have small disparities.

We hope this dataset will be useful for future investigations into various problems including light field synthesis, single view synthesis, and unsupervised geometry learning.

2.5 Synthesizing 4D Ray Depths

We learn the function d to predict depths by minimizing the reconstruction error of the rendered Lambertian light field, along with our novel depth regularization.

Two prominent errors arise when learning to predict depth maps by minimizing the reconstruction error of synthesized views, and we visualize these in Fig. 2.4. In texture-less regions, the depth can be incorrect and depth-based warping will still synthesize the correct image. Therefore, the minimization in Eq. 2.3 has no incentive to predict the correct depth. Second, depths for scene points that are occluded from the input view are also typically incorrect, because predicting the correct depth would cause the synthesized view to sample pixels from the occluder.

Incorrect depths are fine if we only care about the synthesized views. However, the quality of these depths must be improved to consider light field synthesis as an unsupervised learning algorithm to infer depth from a single 2D image. It is difficult to capture large datasets of ground-truth depths for real scenes, especially outdoors, while it is much easier to use capture scenes with a plenoptic camera. We believe that light field synthesis is a promising way to train algorithms to estimate depths from a single image, and we present a strategy to address these depth errors.

We predict depths for every ray in the light field, which is equivalent to predicting a depth map for each view. This enables us to introduce a novel regularization that encourages the predicted depths to be consistent across views and accounts for occlusions, which is a light field generalization of the left-right consistency used in methods such as [40, 168]. Essentially, depths should be consistent for rays coming from the same scene points, which means that the ray depths should be consistent along lines with the same slope:

$$D(\mathbf{x}, \mathbf{u}) = D(\mathbf{x} + \mathbf{k}D(\mathbf{x}, \mathbf{u}), \mathbf{u} - \mathbf{k}) \tag{2.4}$$

for any continuous value of $\mathbf{k}$, as illustrated in Fig. 2.4.

To regularize the predicted depth maps, we minimize the L_1 norm of finite-difference gradients along these sheared lines by setting $k = 1$, which both encourages the predicted depths to be consistent across views and encourages occluders to be sparse:

$$\psi_c(D(\mathbf{x}, \mathbf{u})) = ||D(\mathbf{x}, \mathbf{u}) - D(\mathbf{x} + D(\mathbf{x}, \mathbf{u}), \mathbf{u} - 1)||_1 \tag{2.5}$$

where ψ_c is the consistency regularization loss for predicted ray depths $D(\mathbf{x}, \mathbf{u})$.

Benefits of this regularization are demonstrated in Fig. 2.5. It encourages consistent depths in texture-less areas as well as for rays occluded from the input view,

because predicting the incorrect depths would result in higher gradients along sheared lines as well as new edges in the ray depths.

We additionally use total variation regularization in the spatial dimensions for the predicted depth maps, to encourage them to be sparse in the spatial gradient domain:

$$\psi_{tv}(D(\mathbf{x},\mathbf{u})) = ||\nabla_{\mathbf{x}} D(\mathbf{x},\mathbf{u})||_1 \tag{2.6}$$

Depth Estimation Network We model the function d to estimate 4D ray depths from the input view as a CNN. We use dilated convolutions [160], which allow the receptive field of the network to increase exponentially as a function of the network depth. Hence, each of the predicted ray depths has access to the entire input image without the resolution loss caused by spatial downsampling or pooling. Every convolution layer except for the final layer consists of a 3x3 filter, followed by batch normalization [56] and an exponential linear unit activation function (ELU) [16]. The last layer is followed by a scaled tanh activation function instead of an ELU to constrain the possible disparities to $[-16, 16]$ pixels.

2.6 Synthesizing the 4D Light Field

2.6.1 Lambertian Light Field Rendering

We render an approximate Lambertian light field by using the predicted depths to warp the input view as:

$$L_r(\mathbf{x},\mathbf{u}) = L(\mathbf{x} + \mathbf{u}D(\mathbf{x},\mathbf{u}), \mathbf{0}) \tag{2.7}$$

where $D(\mathbf{x},\mathbf{u})$ is the predicted depth for each ray in the light field. Figure 2.4 illustrates this relationship.

This formulation amounts to using the predicted depths for each ray to render the 4D light field by sampling the input central view image. Since our depth regularization encourages the ray depths to be consistent across views, this effectively encourages different views of the same scene point to sample the same pixel in the input view, resulting in a Lambertian approximation to the light field.

2.6.2 Occlusions and Non-Lambertian Effects

Although predicting a depth for each ray, combined with our depth regularization, allows the network to learn to model occlusions, the Lambertian light fields rendered using these depths are not able to correctly synthesize the values of rays that are occluded from the input view, as demonstrated in Fig. 2.1. Furthermore, this depth-based rendering is not able to accurately predict non-Lambertian effects.

We model the function o to predict occluded rays and non-Lambertian effects as a residual block [44]:

$$o(L_r(\mathbf{x}, \mathbf{u}), D(\mathbf{x}, \mathbf{u})) = \tilde{o}(L_r(\mathbf{x}, \mathbf{u}), D(\mathbf{x}, \mathbf{u})) + L_r(\mathbf{x}, \mathbf{u}) \tag{2.8}$$

where $\tilde{o}$ is modeled as a 3D CNN. We stack all sub-aperture images along one dimension and use a 3D CNN so each filter has access to every 2D view. This 3D CNN predicts a residual that, when added to the approximate Lambertian light field, best predicts the training example true light fields. Structuring this network as a residual block ensures that decreases in the loss are driven by correctly predicting occluded rays and non-Lambertian effects. Additionally, by providing the predicted depths, this network has the information necessary to understand which rays in the approximate light field are incorrect due to occlusions. Figure 2.8 quantitatively demonstrates that this network improves the reconstruction error of the synthesized light fields.

We simply concatenate the estimated depths to the Lambertian approximate light field as the input to a 3D CNN that contains 5 layers of 3D convolutions with 3x3x3 filters (height x width x color channels), batch normalization, and ELU activation functions. The last convolutional layer is followed by a tanh activation function instead of an ELU, to constrain the values in the predicted light field to $[-1, 1]$.

2.6.3 Training

We generate training examples by randomly selecting 192x192x8x8 crops from the training light fields, and spatially downsampling them to 96x96x8x8. We use bilinear interpolation to sample the input view for the Lambertian depth-based rendering, so our network is fully differentiable. We train our network end-to-end using the first-order Adam optimization algorithm [67] with default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$, a learning rate of 0.001, a minibatch size of 4 examples, and depth regularization parameters $\lambda_c = 0.005$ and $\lambda_{tv} = 0.01$.

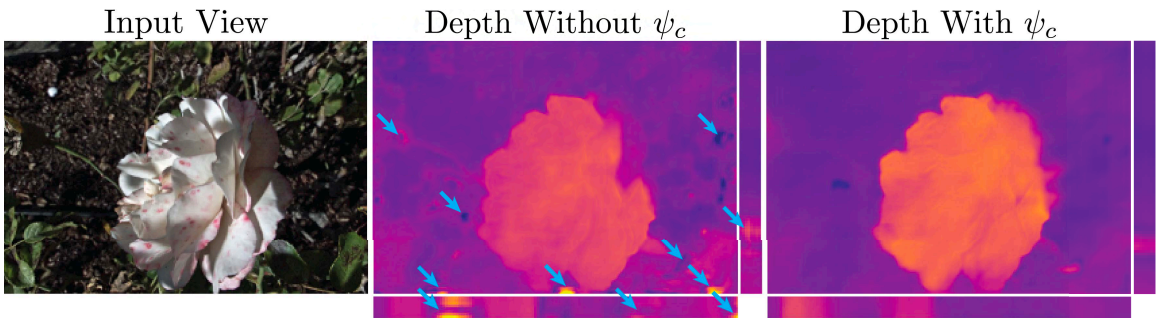


Figure 2.5. Our proposed phyiscally-based depth consistency regularization produces higher-quality estimated depths. Here, we visualize example sub-aperture depth maps where our novel regularization improves the estimated depths for texture-less regions. Blue arrows indicate incorrect depths and depths that are inconsistent across views, as shown in the epipolar slices.

2.7 Results

We validate our light field synthesis algorithm using our testing dataset, and demonstrate that we are able to synthesize compelling 4D ray depths and light fields with complex occlusions and relative depths. No other methods have attempted to synthesize a full 4D light field or 4D ray depths from a single 2D image, so we separately compare our estimated depths to a state-of-the-art light field depth estimation algorithm and our synthesized light fields to a state-of-the-art view synthesis method.

Depth Evaluation We compare our predicted depths to Jeon *et al*. [58], which is a physically-based non-learning depth estimation technique. Note that their algorithm uses the entire ground-truth light field to estimate a 2D depth map, while our algorithm estimates 4D ray depths from a single 2D image. Figure 2.6 qualitatively demonstrates that our unsupervised depth estimation algorithm produces results that are comprable to Jeon *et al*., and even more detailed in many cases.

Synthesized Light Field Evaluation We compare our synthesized light fields to the alternative of using the appearance flow method [167], a state-of-the-art view synthesis method that predicts a flow field to warp an input image to an image from a novel viewpoint. Other recent view synthesis methods are designed for predicting a held-out image from a stereo pair, so it is unclear how to adapt them to predict

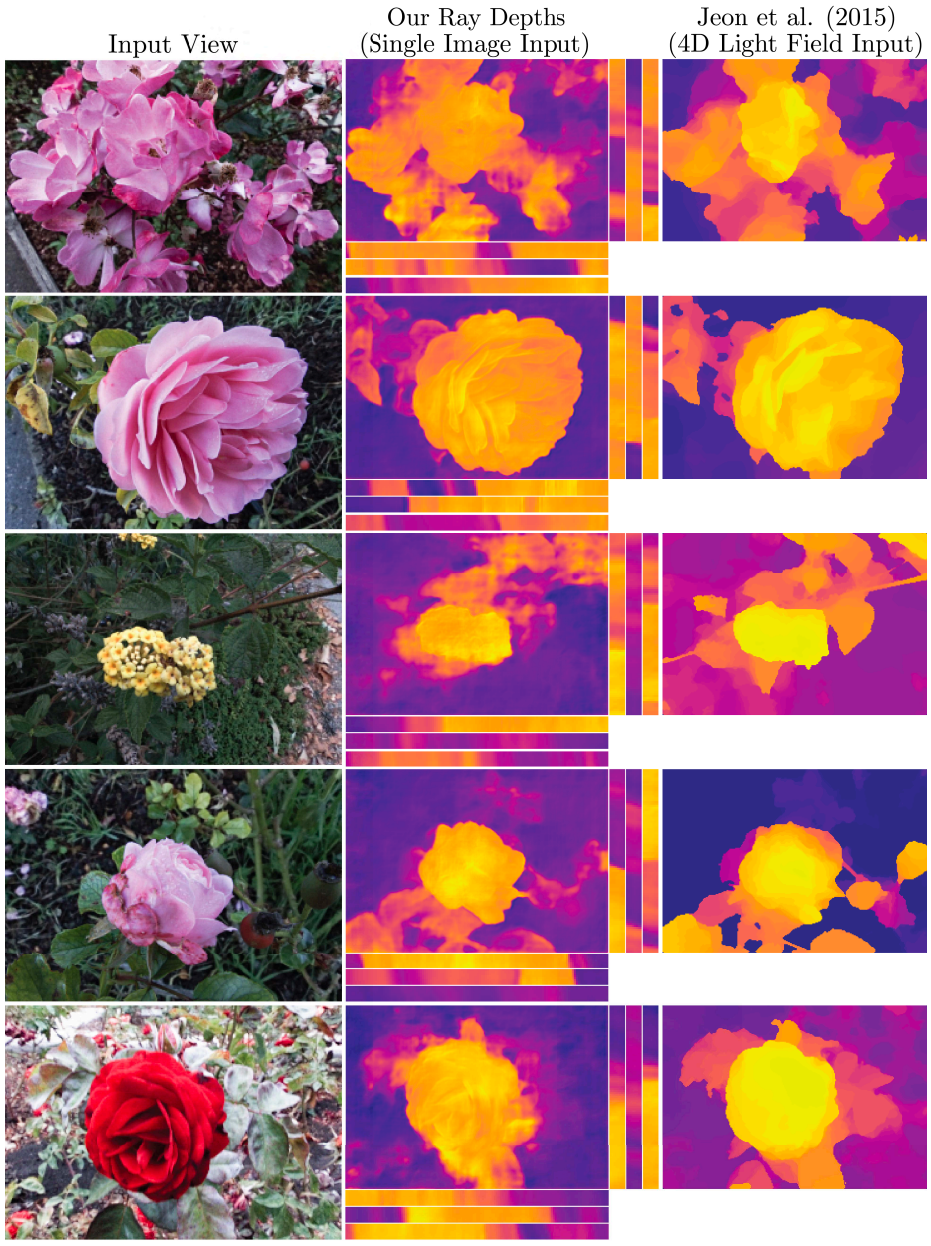


Figure 2.6. We validate our ray depths against the state-of-the-art light field depth estimation. We give Jeon *et al*. [58] a distinct advantage by providing them a ground-truth 4D light field to predict 2D depths, while we use a single 2D image to predict 4D depths. Our estimated depths are comprable, and in some cases superior, to their estimated depths, as shown by the detailed varying depths of the flower petals, leaves, and fine stem structures.

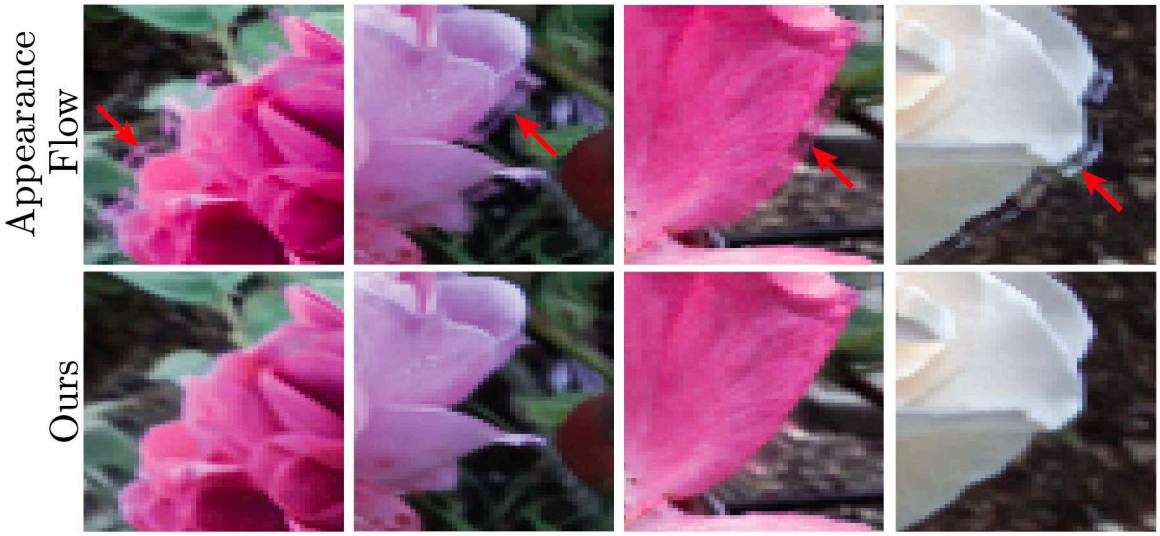


Figure 2.7. We compare our synthesized light fields to the appearance flow method [167]. Qualitatively, appearance flow has difficulties correctly predicting rays occluded from the input view, resulting in artifacts around the edges of the flowers. These types of edge artifacts are highly objectionable perceptually, and the improvement provided by our algorithm subjectively exceeds the quantitative improvement given in Fig. 2.8.

a 4D light field. On the other hand, it is straightforward to adapt the appearance flow method to synthesize a full 4D light field by modifying our depth estimation network to instead predict x and y flow fields to synthesize each sub-aperture image from the input view. We train this network on our training dataset. While appearance flow can be used to synthesize a light field, it does not produce any explicit geometry representation, so unlike our method, appearance flow cannot be used as a strategy for unsupervised geometry learning from light fields.

Figure 2.7 illustrates that appearance flow has trouble synthesizing rays occluded from the input view, resulting in artifacts around occlusion boundaries. Our method is able to synthesize plausible occluded rays and generate convincing light fields. Intuitively, the correct strategy to flow observed rays into occluded regions will change dramatically for flowers with different colors and shapes, so it is difficult to learn. Our approach separates the problems of depth prediction and occluded ray prediction, so the depth prediction network can focus on estimating depth correctly without needing to correctly predict all occluded rays.

To quantitatively evaluate our method, we display histograms for the mean L_1

error on our test dataset for our predicted light fields, our Lambertian light fields, and the appearance flow light fields in Fig. 2.8. We calculate this error over the outermost generated views, since these are the most difficult to synthesize from a central input view. Our predicted light fields have the lowest mean error, and both our predicted and Lambertian approximate light fields have a lower mean error than the appearance flow light fields. We also plot the mean L_1 error as a function of the view position in u, and show that while all methods are best at synthesizing views close to the input view ($(u, v) = 0$), both our predicted and Lambertian light fields consistently outperform the light fields generated by appearance flow. We also tested a CNN that directly regresses from an input image to an output light field, and found that our model outperforms this network with a mean L_1 error of 0.026 versus 0.031 across all views.

Encouragingly, our single view light field synthesis method performs only slightly worse than the light field interpolation method of [61] that takes 4 corner views as input, with a mean L1 error of 0.0176 compared to 0.0145 for a subset of output views not input to either method.

Figure 2.9 displays example light fields synthesized by our method, and demonstrates that we can use our synthesized light fields for photographic effects. Our algorithm is able to predict convincing light fields with complex occlusions and depth ranges, as visualized in the epipolar slices. Furthermore, we can produce realistic photography effects, including extending the aperture from $f/28$ (aperture of the input view) to $f/3.5$ for synthetic defocus blur, and refocusing the full-aperture image from the flower to the background.

Finally, we note that inference is fast, and it takes under 1 second to synthesize a 187x270x8x8 light field and ray depths on a machine with a single Titan X GPU.

Generalization Figure 2.10 demonstrates our method's ability to generalize to input images from a cell phone camera. We show that we can generate convincing ray depths, a high-quality synthesized light field, and interesting photography effects from an image taken with an iPhone 5s.

Finally, we investigate our framework's ability to generalize to other scene classes by collecting a second dataset, consisting of 4281 light fields of various types of toys including cars, figurines, stuffed animals, and puzzles. Figure 2.11 displays an example result from the test set of toys. Although our performance on toys is quantitatively similar to our performance on flowers (the mean L_1 error on the test dataset over all views is 0.027 for toys and 0.026 for flowers), we note that the toys results are perceptually not quite as impressive. The class of toys is much more di-

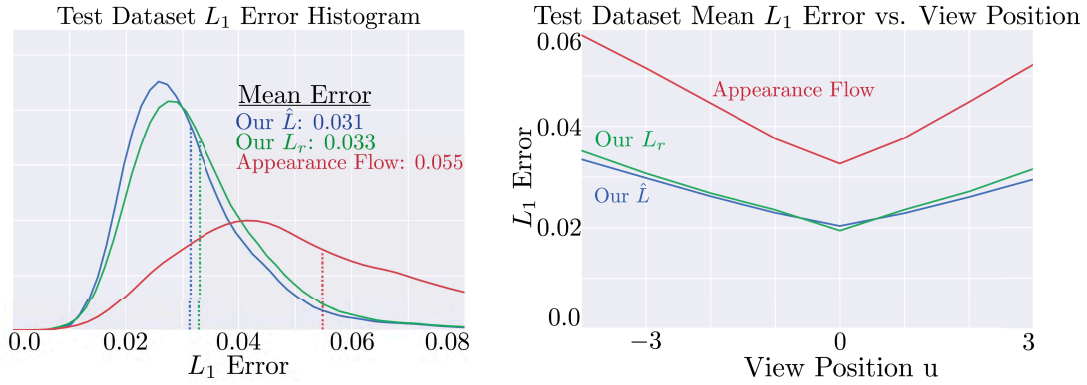


Figure 2.8. To quantitatively validate our results, we visualize histograms of the L_1 errors on the testing dataset for the outermost views of our predicted light fields $\hat{L}$, our Lambertian light fields L_r, and the light fields predicted by appearance flow. Our predicted light fields and Lambertian light fields both have lower errors than those of appearance flow. We also compute the mean L_1 errors as a function of view position u, and demonstrate that our algorithm consistently outperforms appearance flow.

verse than that of flowers, and this suggests that a larger and more diverse dataset would be useful for this scene category.

2.8 Discussion

We have shown that consumer light field cameras enable the practical capture of datasets large enough for training machine learning algorithms to synthesize local light fields of specific scenes from single photographs. It is viable to extend this approach to other niches, as we demonstrate with toys, but it is an open problem to generalize this to the full diversity of everyday scenes. We believe that our work opens up two exciting avenues for future exploration. First, light field synthesis is an exciting strategy for unsupervised geometry estimation from a single image, and we hope that our dataset and algorithm enable future progress in this area. In particular, the notion of enforcing consistent geometry for rays that intersect the same scene point can be used for geometry representations other than ray depths, including voxels, point clouds, and meshes. Second, synthesizing dense light fields is important for capturing VR/AR content, and we believe that this work enables future

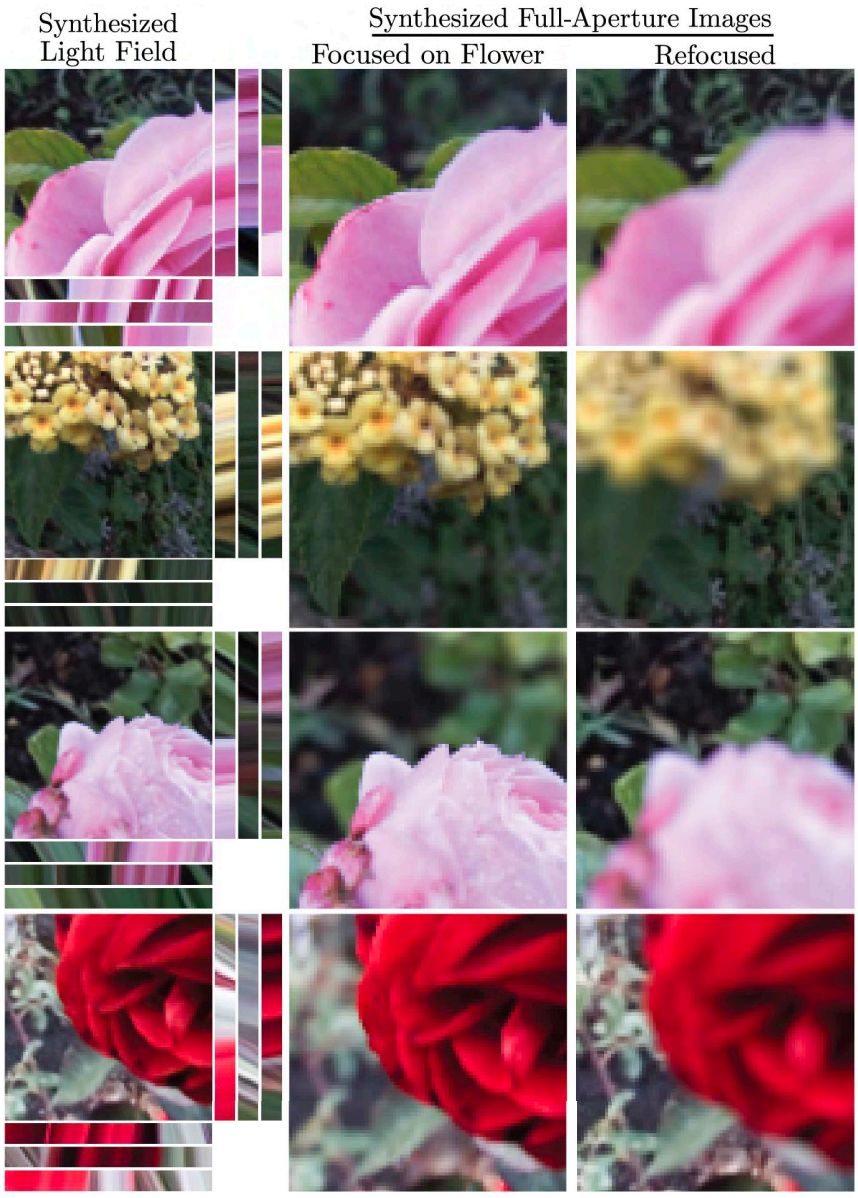


Figure 2.9. We visualize our synthesized light fields as a corner view crop, along with several epipolar slice crops. The epipolar slices demonstrate that our synthesized light fields contain complex occlusions and relative depths. We additionally demonstrate that our light field generated from a single 2D image can be used for synthetic defocus blur, increasing the aperture from $f/28$ to $f/3.5$. Moreover, we can use our light fields to convincingly refocus the full-aperture image from the flowers to the background.

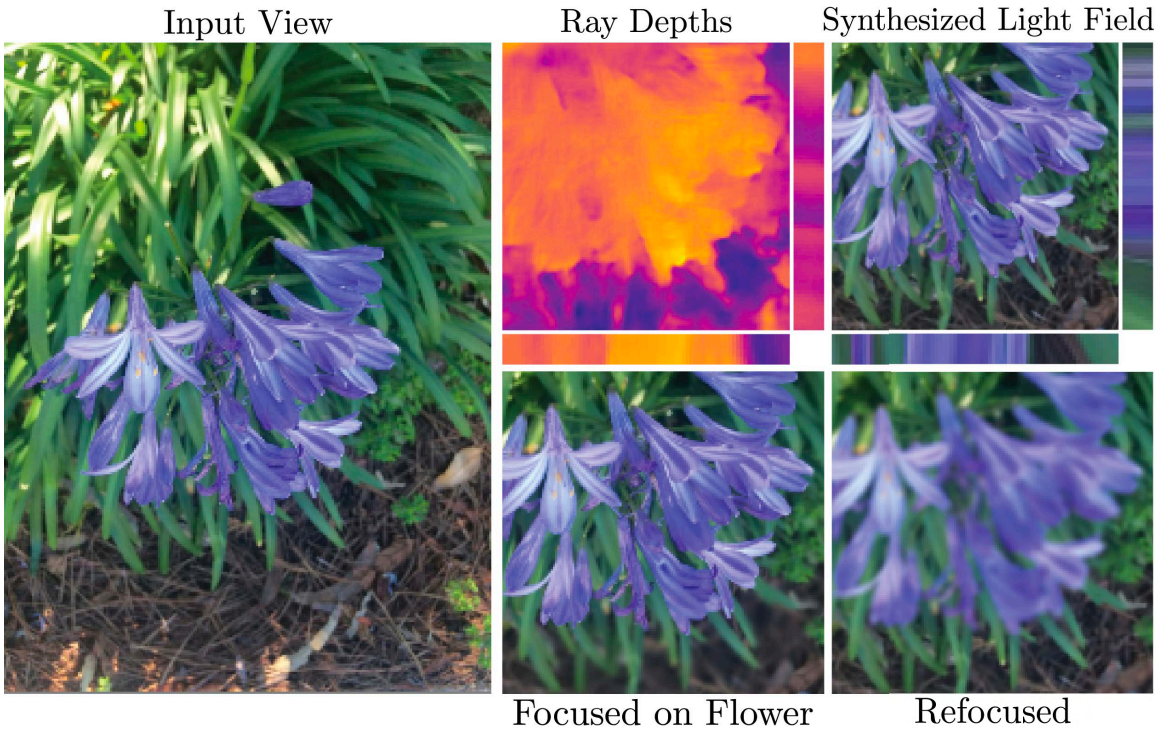


Figure 2.10. Our pipeline applied to cell phone photographs. We demonstrate that our network can generalize to synthesize light fields from pictures taken with an iPhone 5s. We synthesize realistic depth variations and occlusions, as shown in the epipolar slices. Furthermore, we can synthetically increase the iPhone aperture size and refocus the full-aperture image.

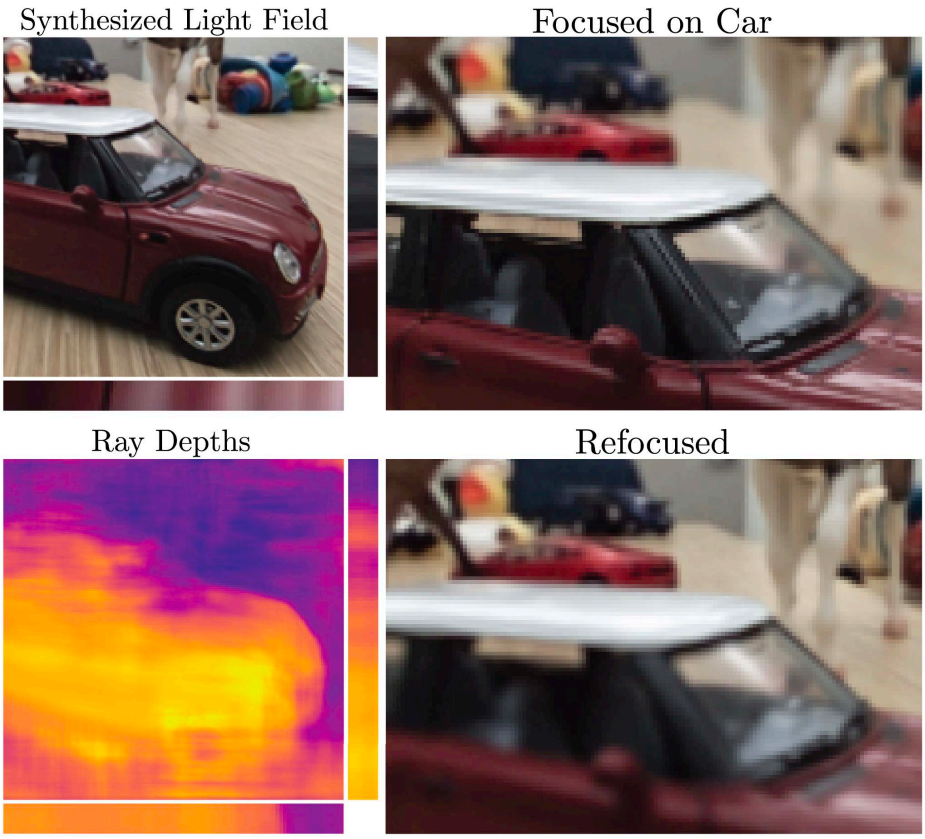


Figure 2.11. We demonstrate that our approach can generalize to scenes of toys, and we display an example test set result.

progress towards generating immersive VR/AR content from sparsely-sampled images.

In summary, this chapter has demonstrated that simply training a deep network to map from an input image to output images without enforcing that all rendered views are related through the same scene geometry is a viable approach for synthesizing local views nearby the input viewpoint. However, this approach struggles as we attempt to synthesize novel views further away from the observed viewpoints. Indeed, Figure 2.8 shows how the error in synthesized images increases as the rendered viewpoints move further away from the observed view. Furthermore, the results presented in Chapter 4 (specifically the results of the "BW Deep" method in Figure 4.9) directly show that this strategy training a deep network to separately map from input views to each output view without considering that all output views should be consistent with a global model of scene geometry results in perceptually-jarring "flickering" artifacts where scene content pops in and out of the synthesized rendered views. In the following chapters, we present strategies that remedy this issue by using fixed 3D scene representations within deep learning pipelines for view synthesis. Indeed, recent work by Tucker and Snavely [140] has shown that using the multiplane image volumetric 3D scene representation we discuss in the following chapters produces state-of-the-art results on the single image view synthesis task investigated in this chapter.

Chapter 3

Local View Extrapolation with Multiplane Images

In this chapter, we focus on a similar problem as we did in the previous chapter: synthesizing novel views in a local neighborhood around an input view, but we take a completely different approach towards the problem. Instead of synthesizing each novel view by separately mapping a set of input views to the target output viewpoint, we directly use the fact that all output views are related through a single 3D representation of the scene's geometry. We specifically investigate the use of multiplane images as a volumetric 3D scene representation. As discussed in Section 3.3.1, multiplane images enjoy the optimization benefits of volumetric scene representations, and are parameter-efficient in that they allocate samples in a manner specifically designed for synthesizing novel "forwards-facing" views of the scene.

Figure 3.1. Given two input images taken from nearby viewpoints, our algorithm predicts an MPI scene representation that can render view extrapolations with disocclusions. Our model improves upon prior work in two specific ways: 1) We reduce depth discretization artifacts due to insufficient depth sampling, as seen in the red zoom of the wood table. 2) We mitigate the repeated texture artifacts produced by prior methods by predicting plausible hidden scene content, as shown in the blue and green zooms where we predict realistic textures behind the fruit bowl and lamp.

3.1 Introduction

View synthesis, the problem of predicting novel views of a scene from a set of captured images, is a central problem in computer vision and graphics. The ability to render nearby views from a single image or a stereo pair can enable compelling photography effects such as 3D parallax and synthetic defocus blur. Furthermore, given a collection of images of a scene taken from different viewpoints, view synthesis could enable free-viewpoint navigation for virtual and augmented reality.

However, there is still a long way to go. State-of-the-art view synthesis algorithms use their input images to estimate a 3D scene representation, which can then be reprojected to render novel views. This approach works well for content visible in the input images, but the quality of novel views degrades rapidly as the target viewpoint moves further away from the input views, thereby revealing more previously-occluded scene content. In this work, we study the problem of view extrapolation where regions of the rendered images observe disoccluded content, and focus specifically on demonstrating view synthesis from a stereo input.

We build upon a state-of-the-art deep learning approach for view synthesis [166] that predicts a scene representation called a multiplane image (MPI) from an input narrow-baseline stereo pair. An MPI consists of a set of fronto-parallel RGBα planes sampled within a reference view camera frustum, as illustrated by Figure 3.2. Diffuse volumetric scene representations such as the MPI are becoming increasingly popular for view synthesis for a number of reasons: 1) They can represent geometric uncertainty in ambiguous regions as a distribution over depths, thereby trading perceptually-distracting artifacts in those ambiguous regions for a more visually-pleasing blur [74, 103]. 2) They are able to convincingly simulate non-Lambertian effects such as specularities [166]. 3) They are straightforward to represent as the output of a CNN and they allow for differentiable rendering, which enables us to train networks for MPI prediction using only triplets of frames from videos for input and supervision [166]. In this work, we extend the MPI prediction framework to enable rendering high-quality novel views up to $4\times$ further from the reference view than was possible in prior work. Our specific contributions are:

Theoretical analysis of MPI limits (Section 3.3.2). We present a theoretical framework, inspired by Fourier theory of volumetric rendering and light fields, to analyze the limits of views that can be rendered from diffuse volumetric representations such as the MPI. We show that the extent of renderable views is limited by the MPI's disparity sampling frequency, even for content visible in both the input and

This chapter is based on joint work published at CVPR 2019 [132].

rendered views, and that this "renderable range" increases linearly with the MPI's disparity sampling frequency.

Improved view extrapolation for visible content (Section 3.3.3). View extrapolation in previous work on MPIs is limited in part by a network architecture that fixes the number of disparity planes during training and testing. Increasing the renderable range of an MPI by simply increasing its fixed number of planes during training is not computationally feasible due to the memory limits of current GPUs. We present a simple solution that increases disparity sampling frequency at test time by replacing the previously used 2D convolutional neural network (CNN) with a 3D CNN architecture and a randomized-resolution training procedure. We demonstrate that this change reduces the depth discretization artifacts found in distant views rendered by prior work, as shown in Figure 3.1.

Predicting disoccluded content for view extrapolation (Section 3.4). We observe and explain why MPIs predicted by prior work [166] contain approximately the same RGB content at each plane, and differ only in α. This behavior results in unrealistic disocclusions with repeated texture artifacts, as illustrated in Figure 3.1. In general, the appearance of hidden scene content is inherently ambiguous, so training a network to simply minimize the distance between rendered and ground truth target views tends to result in unrealistic hallucinations of this occluded content. We propose to improve the realism of predicted disocclusions by constraining the appearance of occluded scene content at every depth to be drawn from visible scene points at or beyond that depth, and present a two-step MPI prediction procedure that enforces this constraint. We demonstrate that this strategy forces predicted disocclusions to contain plausible textures, alleviates the artifacts found in prior work, and produces more compelling extrapolated views than alternative approaches, as illustrated in Figures 3.1 and 3.5.

3.2 Related Work

Traditional approaches for view synthesis. View synthesis is an image-based rendering (IBR) task, with the goal of rendering novel views of scenes given only a set of sampled views. It is useful to organize view synthesis algorithms by the extent to which they use explicit scene geometry [121]. At one extreme are light field rendering [42, 76] techniques, which require many densely sampled input images so that they can render new views by simply slicing the sampled light field without relying on accurate geometry estimation. At the other extreme are techniques such as

view dependent texture mapping that rely entirely on an accurate estimated global mesh and then reproject and blend the texture from nearby input views to render new views [21].

Many successful modern approaches to view synthesis [11, 47, 103, 168] follow a strategy of computing detailed local geometry for each input view followed by forward projecting and blending the local texture from multiple input views to render a novel viewpoint. This research has traditionally focused on interpolating between input views and therefore does not attempt to predict content that is occluded in all input images. In contrast, we focus on the case of view extrapolation, where predicting hidden scene content is crucial for rendering compelling images.

Deep learning approaches for view synthesis. Recently, a promising line of work has focused on training deep learning pipelines end-to-end to render novel views. One class of methods focuses on the challenging problem of training networks to learn about geometry and rendering from scratch and synthesize arbitrarily-distant views from such limited input as a single view [27, 99, 167]. However, the lack of built-in geometry and rendering knowledge limits these methods to synthetic non-photorealistic scenarios.

Other end-to-end approaches have focused on photorealistic view synthesis by learning to model local geometry from a target viewpoint and using this geometry to backwards warp and blend input views. This includes algorithms for interpolating between views along a 1D camera path [31], interpolating between four input corner views sampled on a plane [61], and expanding a single image into a local light field of nearby views (Chapter 2). These methods separately predict local geometry for every novel viewpoint and are not able to guarantee consistency between these predictions, resulting in temporal artifacts when rendering a sequence of novel views. Furthermore, the use of backward projection means that disoccluded regions must be filled in with replicas of visible pixels, so these techniques are limited in their ability to render convincing extrapolated views.

The most relevant methods to our work are algorithms that predict a 3D scene representation from a source image viewpoint and render novel views by differentiably forward projecting this representation into each target viewpoint. This approach ensures consistency between rendered views and allows for the prediction of hidden content. Tulsiani *et al*. and Dhamo *et al*. predict a layered depth image (LDI) representation [22, 141], but this approach is unable to approximate non-Lambertian reflectance effects. Furthermore, training networks to predict LDIs using view synthesis as supervision has proven to be difficult, and the training procedure requires a regularization term that encourages hidden content to resemble occluding content [141], limiting the quality of rendered disocclusions. Zhou *et*

al. proposed the MPI scene representation [166], where novel views are rendered by forward projecting and alpha compositing MPI layers, and a deep learning pipeline is used to train an MPI prediction network using held-out views as supervision. They demonstrated that the MPI scene representation can convincingly render parallax and non-Lambertian effects for a small range of rendered views. We build upon this work and present a theoretical analysis of limits on views rendered from MPIs as well as a new MPI prediction framework that is able to render more compelling view extrapolations with disocclusions.

Inpainting occluded content. Predicting the appearance of content hidden behind visible surfaces can be thought of as 3D scene inpainting. The problem of inpainting in 2D images has an extensive history [43], ranging from early propagation techniques [6] to modern CNN-based inpainting [161]. However, such algorithms must be applied separately to each rendering and therefore do not ensure consistency between different views of the same occluded content.

A few recent works [3, 54, 104, 138] focus on multi-view inpainting, i.e. removing objects and inpainting the resulting empty pixels in a collection of multiple input images. This strategy operates on input image collections instead of scene representations, so it cannot be used to predict occluded content that only appears during view extrapolation.

Finally, a recent line of work [29, 128, 157] focuses on scene shape completion. These methods require an input depth image and only focus on inpainting the shape and semantics of hidden content and not its appearance, so the predicted scenes cannot be used for rendering novel views. In contrast to prior methods, our work addresses the problem of jointly inpainting the geometry, color, and opacity of hidden content in scenes to render convincing disocclusions.

3.3 View Extrapolation for Visible Content

3.3.1 MPI scene representation

The multiplane image (MPI) scene representation, introduced by Zhou *et al*. [166] and illustrated in Figure 3.2, consists of a set of fronto-parallel RGBα planes within a reference camera's view frustum, sampled linearly in disparity (inverse depth). An MPI can be thought of as a frustum-shaped volumetric scene representation where each "voxel" consists of a diffuse RGB color and opacity α. Novel

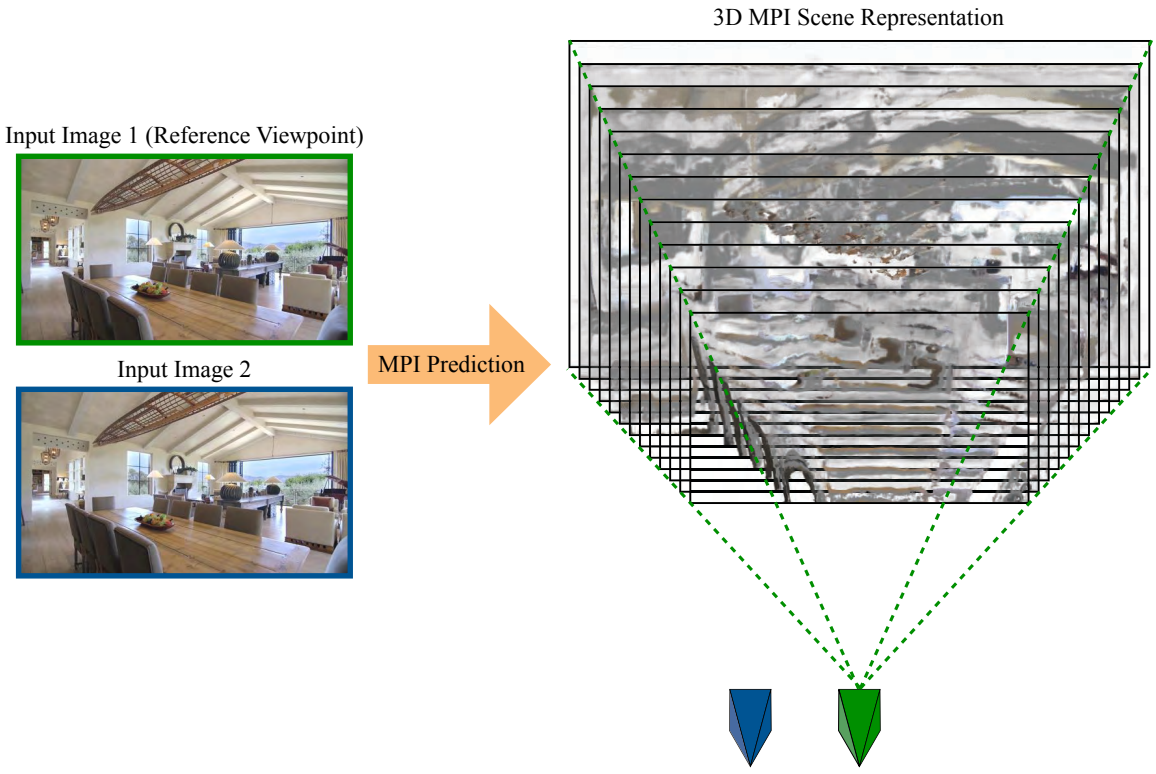


Figure 3.2. **MPI scene representation.** Our work builds on the MPI scene representation and prediction procedure introduced in [166]. We train a deep network that takes two narrow-baseline images of a scene as input (captured at the blue and green camera poses shown above), and predicts an MPI scene representation, consisting of a set of fronto-parallel RGBα planes within a reference camera frustum (signified by the green camera above). Novel views are rendered by alpha compositing along rays from the MPI voxels into the novel viewpoint.

views are rendered from an MPI by alpha compositing the color along rays into the novel view using the "over" operator [74, 105], which is easily implemented as homography-warping each MPI plane onto the sensor plane of the novel view (see Equation 2 in Zhou *et al*. [166]), and alpha compositing the resulting images from back to front.

The MPI scene representation is particularly well-suited for rendering "forwards-facing" views nearby the reference view that face in the same direction. The perspective projection of a scene point onto the image plane moves at a rate relative to the scene point's disparity as the camera translates. Therefore, to support rendering views for a translating camera, the representation's geometry should sample the scene's disparities at the Nyquist rate. The MPI representation fits nicely into this model, as it represents geometry as fronto-parallel planes sampled linearly in disparity. The MPI representation can be considered as a parameter-efficient volumetric representation, since it allocates its discrete volume samples to best support rendering forwards-facing novel views and does not attempt to represent the entire volume at the same level of detail.

3.3.2 Theoretical signal processing limits for rendering visible content

Perhaps surprisingly, there is a limit on views that can be rendered with high fidelity from an MPI, even if we just consider mutually-visible content, i.e., content visible from all input and target viewpoints. Rendering views beyond this limit results in depth discretization artifacts similar to aliasing artifacts seen in volume rendering [74].

We formalize this effect in the context of MPI renderings, and make use of Fourier theory to derive a bound on viewpoints that can be rendered from an MPI with high fidelity. Our model of rendering mutually-visible content from an MPI is conceptually similar to Frequency domain volume rendering [139] using a shear-warp factorization [70]. Additionally, our derivation of an MPI's "renderable range" is inspired by derivations for a 3D display's depth-of-field [169] and light field photography's "refocusable range" [93]. Our main insight is that the 2D Fourier Transform of a view rendered from an MPI can be considered as a 2D slice through the 3D Fourier Transform of the MPI. An MPI is bandlimited by its fixed sampling frequency, so there exists a range of viewpoints outside of which rendered views will have a smaller spatial frequency bandwidth than the input

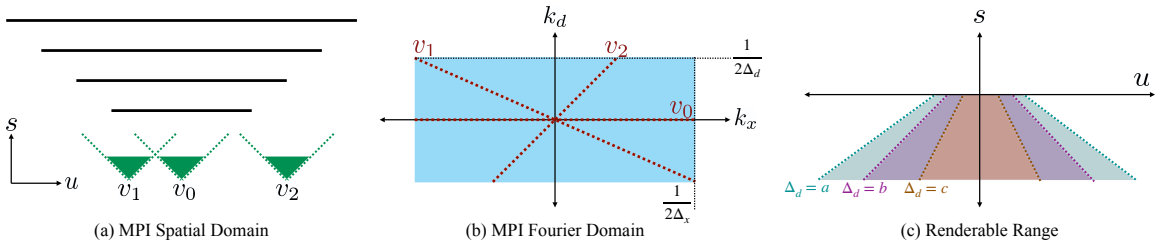


Figure 3.3. **Viewpoint limits for rendering visible content from an MPI.** Views rendered from an MPI without occlusions can be expressed as sheared integral projections of that MPI. (a) Here, we visualize a 2D slice from an MPI, where the y dimension is constant and only the x and z dimensions vary. This MPI is in the reference viewpoint v_0. (b) In the frequency domain, rendered views are equivalent to 1D slices of the 2D MPI spectrum, where views further from the reference viewpoint correspond to Fourier slices at steeper slopes. The MPI spectrum is bandlimited due to its spatial and disparity sampling frequencies, so there is a range of viewpoints outside which rendered views will have a lower spatial bandwidth than the original MPI plane images. Viewpoint v_1 represents the maximum extent of this "renderable range", and v_2 represents a viewpoint outside this range. (c) The renderable range of views is shaped like a truncated cone, and we visualize how the range of renderable views shrinks linearly as we increase the disparity sampling interval Δ_d from $a < b < c$.

images, potentially resulting in aliasing artifacts. We cover the main steps of this derivation below.

Let us consider rendering views from an MPI in the simplified case where (a) the camera is translated but not rotated, and (b) there is no occlusion, so all content is equally visible from every viewpoint. The rendered view $r_{\mathbf{u},s}(\mathbf{x})$ at a lateral translation $\mathbf{u}$ and axial translation s relative to the reference camera center can then be expressed as:

$$r_{\mathbf{u},s}(\mathbf{x}) = \sum_{d \in \mathcal{D}} c(\mathbf{x}', d) = \sum_{d \in \mathcal{D}} c\left(\left(1 - sd\right)\mathbf{x} + \mathbf{u}d, d\right) \tag{3.1}$$

where $c(\mathbf{x}, d)$ is the pre-multiplied RGBα at each pixel coordinate $\mathbf{x}$ and disparity plane d within the set of MPI disparity planes $\mathcal{D}$. Note that $\mathbf{u}$ and s are in units of pixels (such that the camera focal length $f = 1$), and we limit s to the range $-\infty < s < {}^{1}/_{d_{max}}$ because renderings are not defined for viewpoints within the MPI volume. Additionally, note that the disparity d is in units ${}^{1}/_{pixel}$.

To study the limits of views rendered from an MPI, let us consider a worst-case MPI with content in the subset of closest planes, for which we make a locally linear approximation to the coordinate transformation $(\mathbf{x}, d) \rightarrow (\mathbf{x}', d)$:

$$r_{\mathbf{u},s}(\mathbf{x}) = \sum_{d \in \mathcal{D}} c\left(\left(1 - sd_{max}\right)\mathbf{x} + \mathbf{u}d, d\right) \tag{3.2}$$

where d_{max} is a constant. Now, we have expressed the rendering of mutually-visible content as a sheared and dilated integral projection of the MPI. We apply the generalized Fourier slice theorem [93] to interpret the Fourier transform of this integral projection as a 2D slice through the 3D MPI's Fourier transform. The resulting rendered view is the slice's inverse Fourier transform:

$$r_{\mathbf{u},s}(\mathbf{x}) = \mathcal{F}^{-1}\left\{C\left(\frac{k_{\mathbf{x}}}{1 - sd_{max}}, \frac{-\mathbf{u}k_{\mathbf{x}}}{1 - sd_{max}}\right)\right\} \tag{3.3}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform and $C(k_{\mathbf{x}}, k_d)$ is the Fourier transform of $c(\mathbf{x}, d)$.

An MPI is a discretized function, so the frequency support of C lies within a box bounded by $\pm{}^{1}/_{2\Delta_{\mathbf{x}}}$ and $\pm{}^{1}/_{2\Delta_d}$, where $\Delta_{\mathbf{x}}$ is the spatial sampling interval (set by the number of pixels in each RGBα MPI plane image) and Δ_d is the disparity sampling interval (set by the number of MPI planes within the MPI disparity range).

Figures 3.3a and 3.3b illustrate Fourier slices through the MPI's Fourier transform that correspond to rendered views from different lateral positions. Rendered

views further from the reference view correspond to slices at steeper slopes. There is a range of slice slopes within which the spatial bandwidth of the rendered views is equal to that of the MPI, and outside of which the spatial bandwidth of the rendered views decreases linearly with the slice slope.

We can solve for the worst-case "renderable range" by determining the set of slopes whose slices intersect the box in Figure 3.3b at the spatial frequency boundary $\pm 1/2\Delta_{\mathbf{x}}$. This provides constraints on camera positions $(\mathbf{u}, s)$, within which rendered views enjoy the full image bandwidth:

$$s \leq 0, \quad |\mathbf{u}| \leq \frac{\Delta_{\mathbf{x}} \left(1 - s d_{max}\right)}{\Delta_d} \tag{3.4}$$

Figure 3.3c plots the renderable ranges with varying disparity intervals Δ_d, for an MPI with disparities up to d_{max}. The allowed lateral motion extent increases linearly as the target viewpoint moves further axially from the MPI, starting at the reference viewpoint. Decreasing Δ_d linearly increases the amount of allowed lateral camera movement. Intuitively, when rendering views at lateral translations from the reference viewpoint, the renderable range boundary corresponds to views in which adjacent MPI planes are shifted by a single pixel relative to each other before compositing.

3.3.3 Increasing disparity sampling frequency with 3D CNN and randomized-resolution training

Section 3.3.2 establishes that additional MPI planes increases the view extrapolation ability, and that this relationship is linear. Accordingly, the range of extrapolated views rendered by the original MPI method [166] is limited because it uses a 2D CNN to predict a small fixed number of planes (32 planes at a spatial resolution of 1024×576). Simply increasing this fixed number of planes in the network is computationally infeasible during training due to GPU memory constraints. Additionally, training on smaller spatial patches to allow for increased disparity sampling frequency prevents the network from utilizing larger spatial receptive fields, which is important for resolving depth in ambiguous untextured regions.

We propose a simple solution to predict MPIs at full resolution with up to 128 planes at test time by using a 3D CNN architecture, theoretically increasing the view extrapolation ability by $4\times$. The key idea is that because our network is fully 3D convolutional along the height, width, and depth planes dimensions, it can be

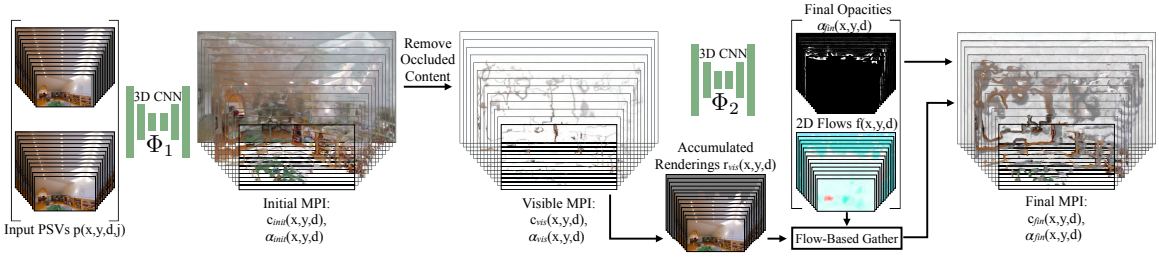


Figure 3.4. **Two-step MPI prediction pipeline.** We propose a two-step procedure to predict convincing hidden content in an MPI for view extrapolation. In the first step, a 3D CNN predicts an initial MPI from the input images' plane-sweep-volumes. Next, occluded content in this MPI is softly removed, resulting in a "first-visible-surface" MPI. In the second step, another 3D CNN predicts final MPI opacities and a 2D flow vector for each MPI voxel. The final MPI RGB colors are computed by using these predicted flows to gather RGB colors from back-to-front cumulative renderings of the visible content. This encourages hidden content at any depth to be synthesized by copying textures of visible content at or behind the same depth, which reduces the output space uncertainty for hidden content and thereby enables convincing view extrapolation with realistic disocclusions.

trained on inputs with varying height, width, and number of depth planes. We use training examples across a spectrum of MPI spatial and disparity sampling frequencies that fit in GPU memory, ranging from MPIs with low spatial and high disparity sampling frequency (128 planes) to MPIs with high spatial and low disparity sampling frequency (32 planes). Perhaps surprisingly, we find that the trained network learns to utilize a receptive field equal to the maximum number of spatial and disparity samples it sees during training, even though no individual training example is of that size.

Our MPI prediction network takes as input a plane-sweep-volume tensor of size $[H, W, D, 3N]$, where H and W are the image height and width, $D = |\mathcal{D}|$ is the number of disparity planes, and N is the number of input images ($N = 2$ in our experiments). This tensor is created by reprojecting each input image to disparity planes $\mathcal{D}$ in a reference view frustum. We use a 3D encoder-decoder network with skip connections and dilated convolutions [160] in the network bottleneck, so that the network's receptive field can encompass the maximum spatial and disparity sampling frequencies used during training.

3.4 View Extrapolation for Hidden Content

In the previous section, we described how view extrapolation is limited by the disparity sampling frequency, which is a fundamental property of the MPI scene representation. View extrapolation is also limited by the quality of hidden content, which is instead a property of the MPI prediction model. Models that train a CNN to directly predict an MPI from an input plane-sweep-volume (which contains homography-warped versions of the same RGB content at each plane) learn the undesirable behavior of predicting approximately the same RGB content at each MPI plane with variation only in α (see Figure 5 in Zhou *et al*. [166]). We observe that this behavior is consistent for models that use either the original 2D CNN architecture or our 3D CNN architecture (Section 3.3.3). Copies of the same RGB content at different MPI layers lead to "repeated texture" artifacts in extrapolated views, where disoccluded content contains repeated copies of the occluder, as visualized in Figure 3.1.

We believe that this undesirable learned behavior is due to both the inductive bias of CNNs that directly predict an MPI from a plane-sweep-volume and the output uncertainty for disocclusions. The probability distribution over hidden scene content, conditioned on observed content, is highly multimodal—there may be many highly plausible versions of the hidden content. As a result, training a network to minimize the distance between rendered and ground truth views produces unrealistic predictions of disocclusions that are some mixture over the space of possible outputs.

We propose to reduce the output uncertainty by constraining the predicted hidden content at any depth, such that its appearance is limited to re-using visible scene content at or behind that depth. This effectively forces the network to predict occluded scene content by copying textures and colors from nearby visible background content. One possible limitation is that this constraint will have difficulty predicting the appearance of self-occlusions where an object extends backwards perpendicular to the viewing direction. However, as argued by the generic viewpoint assumption [32], it is unlikely that our reference viewpoint happens to view an object exactly at the angle at which it extends backwards along the viewing direction. In general, the majority of disoccluded pixels view background content instead of self-occlusions.

We enforce our constraint on the appearance of occluded content with a two-step MPI prediction procedure. The first step provides an initial estimate of the geometry and appearance of scene content visible from the reference viewpoint.

The second step uses this to predict a final MPI where the color at each voxel is parameterized by a flow vector that points to a visible surface's color to copy.

In the first step, an input plane-sweep volume p is constructed by reprojecting j input images $i_{\mathbf{v}_j}$, each captured at a viewpoint $\mathbf{v}_j$, to disparity planes $d \in \mathcal{D}$. The 3D CNN Φ_1 of Section 3.3.3 takes this plane-sweep volume and predicts an initial MPI's RGB and α values, c_{init} and α_{init}:

$$c_{init}(x, y, d), \alpha_{init}(x, y, d) = \Phi_1\big(p(x, y, d, j)\big). \tag{3.5}$$

This initial MPI typically contains repeated foreground textures in occluded regions of the scene. In the second step of our procedure, we aim to preserve the predicted geometry and appearance of the first visible surface from the initial MPI while re-predicting the appearance and geometry of hidden content and enforcing our flow-based appearance constraint. We softly remove hidden RGB content from this initial MPI by multiplying each MPI RGB value by its transmittance t relative to the reference viewpoint $\mathbf{v}_0$:

$$t_{\mathbf{v}_0}(x, y, d) = \alpha_{init}(x, y, d) \prod_{d'>d} [1 - \alpha_{init}(x, y, d')] \tag{3.6}$$

$$\begin{aligned} c_{vis}(x, y, d) &= c_{init}(x, y, d) t_{\mathbf{v}_0}(x, y, d) \\ \alpha_{vis}(x, y, d) &= t_{\mathbf{v}_0}(x, y, d) \end{aligned} \tag{3.7}$$

where c_{vis} and α_{vis} are the MPI RGBα planes from which content that is occluded from the reference view has been softly removed. Intuitively, a voxel's transmittance (Equation 3.6) describes the extent to which an MPI voxel's color contributes to the rendered reference view.

A second CNN Φ_2 takes this reference-visible MPI, consisting of c_{vis} and α_{vis}, as input and predicts opacities $\alpha_{fin}(x, y, d)$ and a 2D flow vector for each MPI voxel $f(x, y, d) = [f_x(x, y, d), f_y(x, y, d)]$:

$$\alpha_{fin}(x, y, d), f(x, y, d) = \Phi_2\big(c_{vis}(x, y, d), \alpha_{vis}(x, y, d)\big). \tag{3.8}$$

The final MPI's colors $c_{fin}(x, y, d)$ are computed by using these predicted flows to gather colors from renderings of the visible content at or behind each plane $r_{vis}(x, y, d)$:

$$r_{vis}(x, y, d) = \sum_{d' \leq d} \big[c_{vis}(x, y, d')\big] \tag{3.9}$$

$$c_{fin}(x, y, d) = r_{vis}\left(x + f_x(x, y, d), y + f_y(x, y, d), d\right).$$

We gather the color from r_{vis} using bilinear interpolation for differentiability. This constraint restricts the appearance of hidden content at each depth to be drawn from visible scene points at or beyond that depth.

Algorithm	SSIM_{fov}	SSIM_{occ}	NAT_{occ}
Original MPI [166]	0.838	0.803	0.805
Our r_{init}	0.858	0.811	0.904
r_{init} + Adversarial Disocclusion	0.853	0.791	0.849
Disocclusion Inpainting [161]	0.808	0.691	0.227
Our r_{fin}	0.853	0.814	0.931

Table 3.1. **Quantitative evaluation.** Images rendered from our predicted MPIs are quantitatively superior to those rendered from the original MPI model [166]. Furthermore, our method predicts disocclusions that are both closer to the ground truth hidden content and more perceptually plausible than alternative methods.

3.5 Training Loss

As in Zhou *et al*. [166], we train our MPI prediction pipeline using view synthesis as supervision. Our training loss is simply the sum of reconstruction losses for rendering a held-out novel view r_{gt} at target camera pose $\mathbf{v}_t$, using both our initial and final predicted MPIs. These MPIs are predicted from input images $i_{\mathbf{v}_0}$ and $i_{\mathbf{v}_1}$. We use a deep feature matching loss $\mathcal{L}_{VGG}$ for layers from the VGG-19 network [123], using the implementation of Chen and Koltun [12]. The total loss $\mathcal{L}$ for each training example is:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{VGG}(r_{init}(i_{\mathbf{v}_0}, i_{\mathbf{v}_1}, \mathbf{v}_t), r_{gt}) + \\ & \mathcal{L}_{VGG}(r_{fin}(i_{\mathbf{v}_0}, i_{\mathbf{v}_1}, \mathbf{v}_t), r_{gt})\end{aligned} \tag{3.10}$$

where r_{init} and r_{fin} are rendered views from the initial and final predicted MPIs.

3.6 Results

The following section presents quantitative and qualitative evidence to validate the benefits of our method.

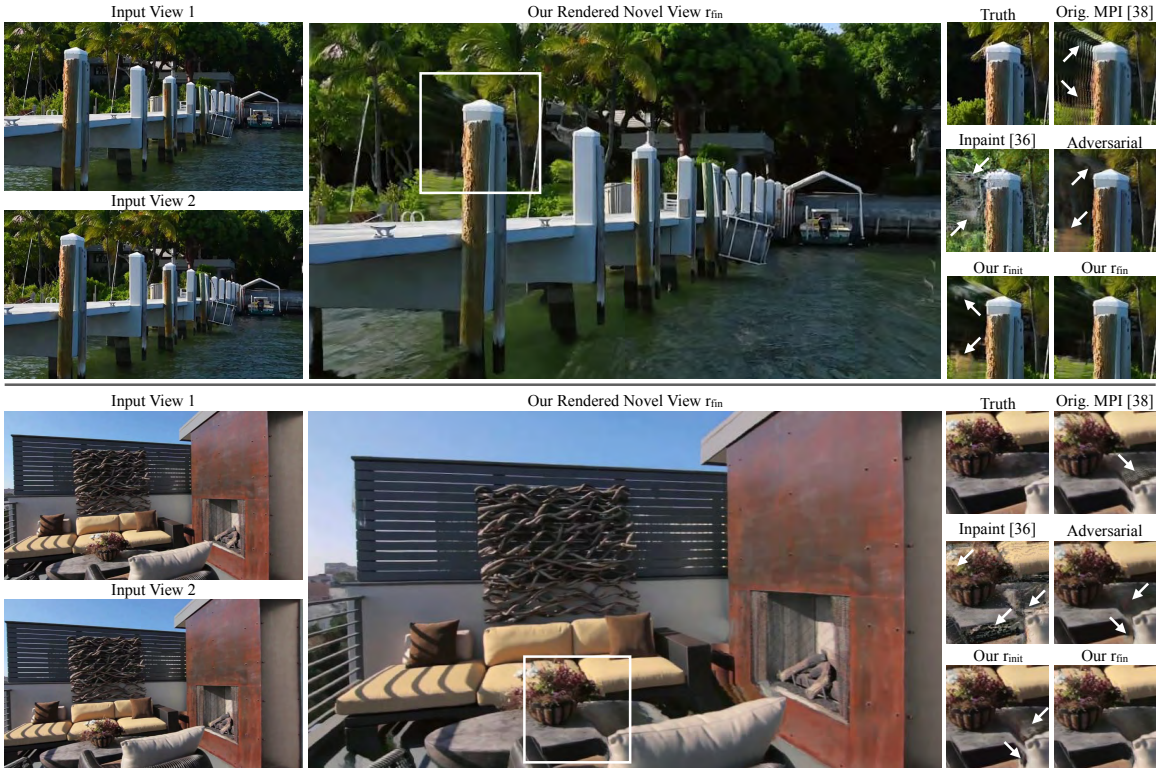


Figure 3.5. **Qualitative comparison of rendered novel views.** Our method predicts MPIs with convincing hidden content, as demonstrated by the disoccluded foliage textures to the left of the wooden pole in the top example, and the disoccluded region to the left of the grey pillow in the bottom example. Renderings from alternative methods contain depth discretization artifacts, implausible colors, blurry textures, and repeated textures in disoccluded regions.

3.6.1 Experiment details

We train and evaluate on the open-source YouTube Real Estate 10K dataset [166], which contains approximately 10,000 YouTube videos of indoor and outdoor real estate scenes along with computed camera poses for each video frame. We generate training examples on the fly by sampling two source frames and a target frame from a randomly chosen video, so that the target image is not in between the source images (and therefore requires view extrapolation, not view interpolation) for ∼87% of the training examples.

`https://google.github.io/realestate10k/`

The dataset is split into 9,000 videos for training and 1,000 for testing, where the test set videos do not overlap with those in the training dataset. From these test videos, we randomly sample 6,800 test triplets, each consisting of two input frames and a single target frame.

3.6.2 Evaluation metrics

We use three metrics for our quantitative comparisons:

SSIM_{fov}: To evaluate the overall quality of rendered images, we use the standard SSIM [146] metric computed over the region of the target image that views all MPI planes.

SSIM_{occ}: To specifically assess the accuracy of predicted disocclusions, we evaluate SSIM over the subset of pixels that were not visible from the input reference viewpoint. We determine whether a pixel in a rendered target image is disoccluded by examining the MPI voxels that contribute to the rendered pixel's value, and thresholding the maximum change in transmittance of these contributing voxels between the reference and target viewpoint. Similarly to Equation 3.6, we can compute the transmittance of each MPI voxel from a target viewpoint $\mathbf{v}_t$ as:

$$t_{\mathbf{v}_t}(x, y, d) = \alpha_{\mathbf{v}_t}(x, y, d) \prod_{d' > d} [1 - \alpha_{\mathbf{v}_t}(x, y, d')] \tag{3.11}$$

where $\alpha_{\mathbf{v}_t}$ is an MPI α plane homography-warped onto the sensor plane of viewpoint $\mathbf{v}_t$. We consider a pixel (x, y) in the target rendered view as a member of the disoccluded pixels set $\mathcal{H}$ if the transmittance t of any contributing MPI voxel is some threshold value greater than the same voxel's transmittance when rendering the reference viewpoint:

$$\mathcal{H} = \left\{ (x, y) : \max_d \left(t_{\mathbf{v}_t}(x, y, d) - t_{\mathbf{v}_0 \to \mathbf{v}_t}(x, y, d) \right) \geq \epsilon \right\} \tag{3.12}$$

where $t_{\mathbf{v}_0 \to \mathbf{v}_t}$ is the transmittance relative to the reference viewpoint, warped into the target viewpoint so that both transmittances are in the same reference frame. We compute disoccluded pixels using α_{init} for all models, to ensure that each model is evaluated on the same set of pixels. We set $\epsilon = 0.075$ in our experiments.

NAT_{occ}: To quantify the perceptual plausibility of predicted disoccluded content, we evaluate a simple image prior over disoccluded pixels. We use the negative log of the Earth Mover's (Wasserstein-1) distance between gradient magnitude histograms of the rendered disoccluded pixels and the ground-truth pixels in each

target image. Intuitively, realistic rendered image content should have a distribution of gradients that is similar to that of the true natural image [122, 147], and therefore a higher NAT_{occ} score.

3.6.3 Comparison to baseline MPI prediction

We first show that renderings from both our initial and final predicted MPIs (r_{init} and r_{fin}) are superior to those from the original MPI method [166], which was demonstrated to significantly outperform other recent view synthesis methods [61, 164]. The increase in SSIM_{fov} from "Original MPI" (Table 3.1 row 2) to "Our r_{init}" (row 3) demonstrates the improvement from our method's increased disparity sampling frequency. Furthermore, the increase in SSIM_{occ} and NAT_{occ} from "Original MPI" (row 2) to "Our r_{fin}" (row 6) demonstrates that our method predicts disoccluded content that is both closer to the ground truth and more plausible. Figure 3.5 qualitatively demonstrates that renderings from our method contain fewer depth discretization artifacts than renderings from the original MPI work, and that renderings from our final MPI contain more realistic disocclusions without "repeated texture" artifacts.

3.6.4 Evaluation of hidden content prediction

We compare occluded content predicted by our model to the following alternative disocclusion prediction strategies:

Our r_{init}: We first compare renderings "Our r_{fin}" from our full method to the ablation "Our r_{init}", which does not enforce our flow-based occluded content appearance constraint. The improvement in SSIM_{occ} and NAT_{occ} from Table 3.1 row 3 to row 6 and the qualitative results in Figure 3.5 demonstrate that our full method renders disocclusions that are both closer to the ground truth and more perceptually plausible with fewer "repeated texture" artifacts.

"r_{init} + Adversarial Disocclusions": Next, we compare to an alternative two-step MPI prediction strategy. We use an identical Φ_1 to predict the initial MPI, but Φ_2 directly predicts RGBα planes instead of α and flow planes. We apply an adversarial loss to the resulting rendered target image to encourage realistic disocclusions. Table 3.1 row 4 demonstrates that this strategy renders disocclusions that are less accurate but more perceptually plausible than the original MPI method, due to the adversarial loss. However, Figure 3.5 demonstrates that the renderings from

our full method contain sharper content and more accurate colors than those of the "r_{init} + Adversarial Disocclusions" strategy. We hypothesize that this is due to the difficulty of training a discriminator network when the number and location of "fake" disoccluded pixels varies drastically between training examples.

"Disocclusion Inpainting": Finally, we compare to an image-based disocclusion prediction strategy. We remove the disoccluded pixels from our final MPI renderings and re-predict them using a state-of-the-art deep learning image inpainting model [161]. Table 3.1 row 5 shows that this strategy results in an overall quality reduction, especially for the accuracy and plausibility of disoccluded regions. Figure 3.5 visualizes the unrealistic inpainting results. Furthermore, as shown in our video, predicting disocclusions separately for each rendered image creates distracting temporal artifacts in rendered camera paths because the appearance of disoccluded content changes with the viewpoint.

3.7 Discussion

We have presented a theoretical signal processing analysis of limits for views that can be rendered from an MPI scene representation, and a practical deep learning method to predict MPIs that theoretically allow for $4\times$ more lateral movement in rendered views than prior work. This improvement is due to our method's ability to predict MPIs with increased disparity sampling frequency and our flow-based hidden content appearance constraint to predict MPIs that render convincing disocclusion effects. However, there is still a lot of room for improvement in predicting scene representations for photorealistic view synthesis that contain convincing occluded 3D content and are amenable to deep learning pipelines, and we hope that this work inspires future progress along this exciting research direction.

In this chapter, we have demonstrated how a parameter-efficient volumetric representation such as a multiplane image can be effective for local view extrapolation. Although this representation can synthesize photorealistic novel views, we have shown that it is only able to do so within a range limited by the number of planes. In the next chapter, we discuss a strategy to extend this range by representing a scene as multiple overlapping multiplane images and intelligently blending between them to render novel views.

Chapter 4

View Interpolation with Multiplane Images

In this chapter, we demonstrate that multiplane images are a compelling volumetric 3D scene representation for synthesizing novel views that interpolate between images captured on an irregular grid-like pattern. This use case is particularly relevant for enabling users to casually capture the appearance of a scene for virtual reality experiences.

4.1 Introduction

The most compelling virtual experiences completely immerse the viewer in a scene, and a hallmark of such experiences is the ability to view the scene from a close interactive distance. This is currently possible with synthetically rendered scenes, but this level of intimacy has been very difficult to achieve for virtual experiences of real world scenes.

Ideally, we could simply sample the scene's light field and interpolate the rel-

This chapter is based on joint work published at SIGGRAPH 2019 [89].

Figure 4.1. We present a simple and reliable method for view synthesis from a set of input images captured by a handheld camera in an irregular grid pattern. We theoretically and empirically demonstrate that our method enjoys a prescriptive sampling rate that requires $4000\times$ fewer input views than Nyquist for high-fidelity view synthesis of natural scenes. Specifically, we show that this rate can be interpreted as a requirement on the pixel-space disparity of the closest object to the camera between captured views (Section 4.3). After capture, we expand all sampled views into layered representations that can render high-quality local light fields. We then blend together renderings from adjacent local light fields to synthesize dense paths of new views (Section 4.4). Our rendering consists of simple and fast computations (homography warping and alpha compositing) that can generate new views in real-time.

evant captured images to render new views. Such light field sampling strategies are particularly appealing because they pose the problem of image-based rendering (IBR) in a signal processing framework where we can directly reason about the density and pattern of sampled views required for any given scene. However, Nyquist rate view sampling is intractable for scenes with content at interactive distances, as the required view sampling rate increases linearly with the reciprocal of the closest scene depth. For example, for a scene with a subject at a depth of 0.5 meters captured by a mobile phone camera with a 64° field of view and rendered at 1 megapixel resolution, the required sampling rate is an intractable 2.5 million images per square meter. Since it is not feasible to capture all the required images, the IBR community has moved towards view synthesis algorithms that leverage geometry estimation to predict the missing views.

State-of-the-art algorithms pose the view synthesis problem as the prediction of novel views from an unstructured set or arbitrarily sparse grid of input camera views. While the generality of this problem statement is appealing, abandoning a plenoptic sampling framework sacrifices the crucial ability to rigorously reason about the view sampling requirements of these methods and predict how their performance will be affected by the input view sampling pattern. When faced with

a new scene, users of these methods are limited to trial-and-error to figure out whether a set of sampled views will produce acceptable results for a virtual experience.

Instead, we propose a view synthesis approach that is grounded within a plenoptic sampling framework and can precisely prescribe how densely a user must capture a given scene for reliable rendering performance. Our method is conceptually simple and consists of two main stages. We first use a deep network to promote each source view to a layered representation of the scene that can render a limited range of views, advancing recent work on the multiplane image (MPI) representation [166]. We then synthesize novel views by blending renderings from adjacent layered representations.

Our theoretical analysis shows that the number of input views required by our method decreases quadratically with the number of planes we predict for each layered scene representation, up to limits set by the camera field of view. We empirically validate our analysis and apply it in practice to render novel views with the same perceptual quality as Nyquist view sampling while using up to $64^2 \approx 4000\times$ fewer images.

It is impossible to break the Nyquist limit with full generality, but we show that it is possible to achieve Nyquist level performance with greatly reduced view sampling by specializing to the subset of natural scenes. This capability is primarily due to our deep learning pipeline, which is trained on renderings of natural scenes to estimate high quality layered scene representations that produce locally consistent light fields.

In summary, our key contributions are:

1. An extension of plenoptic sampling theory that directly specifies how users should sample input images for reliable high quality view synthesis with our method.

2. A practical and robust solution for capturing and rendering complex real world scenes for virtual exploration.

3. A demonstration that carefully crafted deep learning pipelines using local layered scene representations achieve state-of-the-art view synthesis results.

We extensively validate our derived prescriptive view sampling requirements and demonstrate that our algorithm quantitatively outperforms traditional light field reconstruction methods as well as state-of-the-art view interpolation algorithms across a range of sub-Nyquist view sampling rates. We highlight the practi-

cality of our method by developing an augmented reality app that implements our derived sampling guidelines to help users capture input images that produce reliably high-quality renderings with our algorithm. Additionally, we develop mobile and desktop viewer apps that render novel views from our predicted layered representations in real-time. Finally, we qualitatively demonstrate that our algorithm reliably produces state-of-the-art results across a diverse set of complex real-world scenes.

4.2 Related Work

Image-based rendering (IBR) is the fundamental computer graphics problem of rendering novel views of objects and scenes from sampled views. We find that it is useful to categorize IBR algorithms by the extent to which they use explicit scene geometry, as done by Shum and Kang [121].

4.2.1 Plenoptic Sampling and Reconstruction

Light field rendering [76] eschews any geometric reasoning and simply samples images on a regular grid so that new views can be rendered as slices of the sampled light field. Lumigraph rendering [42] showed that using approximate scene geometry can ameliorate artifacts due to undersampled or irregularly sampled views.

The plenoptic sampling framework [9] analyzes light field rendering using signal processing techniques and shows that the Nyquist view sampling rate for light fields depends on the minimum and maximum scene depths. Furthermore, they discuss how the Nyquist view sampling rate can be lowered with more knowledge of scene geometry. Zhang and Chen [162] extend this analysis to show how non-Lambertian and occlusion effects increase the spectral support of a light field, and also propose more general view sampling lattice patterns.

Rendering algorithms based on plenoptic sampling enjoy the significant benefit of prescriptive sampling; given a new scene, it is easy to compute the required view sampling density to enable high-quality renderings. Many modern light field acquisition systems have been designed based on these principles, including large-scale camera systems [150, 98] and a mobile phone app [19].

We posit that prescriptive sampling is necessary for practical and useful IBR

algorithms, and we extend prior theory on plenoptic sampling to show that our deep-learning-based view synthesis strategy can significantly decrease the dense sampling requirements of traditional light field rendering. Our novel view synthesis pipeline can also be used in future light field acquisition hardware systems to reduce the number of required cameras.

4.2.2 Geometry-Based View Synthesis

Many IBR algorithms attempt to leverage explicit scene geometry to synthesize new views from arbitrary unstructured sets of input views. These approaches can be meaningfully categorized as either using global or local geometry.

Techniques that use global geometry generally compute a single global mesh from a set of unstructured input images. Simply texture mapping this global mesh can be effective for constrained situations such as panoramic viewing with mostly rotational and little translational viewer movement [45, 46], but this strategy can only simulate Lambertian materials. Surface light fields [151] are able to render convincing view-dependent effects, but they require accurate geometry from dense range scans and hundreds of captured images to sample the outgoing radiance at points on an object's surface.

Many free-viewpoint IBR algorithms are based upon a strategy of locally texture mapping a global mesh. The influential view-dependent texture mapping algorithm [21] proposed an approach to render novel views by blending nearby captured views that have been reprojected using a global mesh. Work on Unstructured Lumigraph Rendering [7] focused on computing per-pixel blending weights for reprojected images and proposed a heuristic algorithm that satisfied key properties for high-quality rendering. Unfortunately, it is very difficult to estimate high-quality meshes whose geometric boundaries align well with edges in images, and IBR algorithms based on global geometry typically suffer from significant artifacts. State-of-the-art algorithms [47, 48] attempt to remedy this shortcoming with complicated pipelines that involve both global mesh and local depth map estimation. However, it is difficult to precisely define view sampling requirements for robust mesh estimation, and the mesh estimation procedure typically takes multiple hours, making this strategy impractical for casual content capture scenarios.

IBR algorithms that use local geometry [11, 13, 68, 87, 97] avoid difficult and expensive global mesh estimation. Instead, they typically compute detailed local geometry for each input image and render novel views by reprojecting and blending nearby input images. This strategy has also been extended to simulate non-

Lambertian reflectance by using a second depth layer [124]. The state-of-the-art Soft3D algorithm [103] blends between reprojected local layered representations to render novel views, which is conceptually similar to our strategy. However, Soft3D computes each local layered representation by aggregating heuristic measures of depth uncertainty over a large neighborhood of views. We instead train a deep learning pipeline end-to-end to optimize novel view quality by predicting each of our local layered representations from a much smaller neighborhood of views. Furthermore, we directly pose our algorithm within a plenoptic sampling framework, and our analysis directly applies to the Soft3D algorithm as well. We demonstrate that the high quality of our deep learning predicted local scene representations allows us to synthesize superior renderings without requiring the aggregation of geometry estimates over large view neighborhoods, as done in Soft3D. This is especially advantageous for rendering non-Lambertian effects because the apparent depth of specularities generally varies with the observation viewpoint, so smoothing the estimated geometry over large viewpoint neighborhoods prevents accurate rendering of these effects.

Other IBR algorithms [2] have attempted to be more robust to incorrect camera poses or scene motion by interpolating views using more general 2D optical flow instead of 1D depth. Local pixel shifts are also encoded in the phase information, and algorithms have exploited this to extrapolate views from micro-baseline stereo pairs [23, 65, 164] without explicit flow computation. However, these methods require extremely close input views and are not suited for large baseline view interpolation.

4.2.3 Deep Learning for View Synthesis

Other recent methods have trained deep learning pipelines end-to-end for view synthesis. This includes recent angular superresolution methods [152, 159] that interpolate dense views within a light field camera's aperture but cannot handle sparser input view sampling since they do not model scene geometry. The DeepStereo algorithm [31], deep learning based light field camera view interpolation [61], and single view local light field synthesis (Chapter 2) each use a deep network to predict depth separately for every novel view. However, predicting local geometry separately for each view results in inconsistent renderings across smoothly-varying viewpoints.

Finally, Zhou *et al*. [166] introduce a deep learning pipeline to predict an MPI from a narrow baseline stereo pair for the task of stereo magnification. As opposed

to previous deep learning strategies for view synthesis, this approach enforces consistency by using the same predicted scene representation to render all novel views. We adopt MPIs as our local light field representation and introduce specific technical improvements to enable larger-baseline view interpolation from many input views, in contrast to local view extrapolation from a stereo pair using a single MPI. We predict multiple MPIs, one for each input view, and train our system end-to-end through a blending procedure to optimize the resulting MPIs to be used in concert for rendering output views. We propose a 3D convolutional neural network (CNN) architecture that dynamically adjusts the number of depth planes based on the input view sampling rate, rather than a 2D CNN with a fixed number of output planes. Additionally, we show that state-of-the-art performance requires only an easily-generated synthetic dataset and a small real fine-tuning dataset, rather than a large real dataset. This allows us to generate training data captured on 2D irregular grids similar to handheld view sampling patterns, while the YouTube dataset in Zhou *et al*. [166] is restricted to 1D camera paths.

4.3 Theoretical Sampling Analysis

The overall strategy of our method is to use a deep learning pipeline to promote each sampled view to a layered scene representation with D depth layers, and render novel views by blending between renderings from neighboring scene representations. In this section, we show that the full set of scene representations predicted by our deep network can be interpreted as a specific form of light field sampling. We extend prior work on plenoptic sampling to show that our strategy can theoretically reduce the number of required sampled views by a factor of D^2 compared to the number required by traditional Nyquist view sampling. Section 4.6.1 empirically shows that we are able to take advantage of this bound to reduce the number of required views by up to $64^2 \approx 4000\times$.

In the following analysis, we consider a "flatland" light field with a single spatial dimension x and view dimension u for notational clarity, but note that all findings apply to general light fields with two spatial and two view dimensions.

Table 4.1. Reference for symbols used in Section 4.3.

Symbol	Definition
D	Number of depth planes
W	Camera image width (pixels)
f	Camera focal length (meters)
Δ_x	Pixel size (meters)
Δ_u	Baseline between cameras (meters)
K_x	Highest spatial frequency in sampled light field
B_x	Highest spatial frequency in continuous light field
$z_{\min}$	Closest scene depth (meters)
$z_{\max}$	Farthest scene depth (meters)
$d_{\max}$	Maximum disparity between views (pixels)

4.3.1 Nyquist Rate View Sampling

Initial work on plenoptic sampling [9] derived that the Fourier support of a light field, ignoring occlusion and non-Lambertian effects, lies within a double-wedge shape whose bounds are set by the minimum and maximum scene depths $z_{\min}$ and $z_{\max}$, as visualized in Figure 4.2. Zhang and Chen [162] showed that occlusions expand the light field's Fourier support because an occluder convolves the spectrum of the light field due to farther scene content with a kernel that lies on the line corresponding to the occluder's depth. The light field's Fourier support considering occlusions is limited by the effect of the closest occluder convolving the line corresponding to the furthest scene content, resulting in the parallelogram shape illustrated in Figure 4.3a, which can only be packed half as densely as the double-wedge. The required maximum camera sampling interval Δ_u for a light field with occlusions is:

$$\Delta_u \leq \frac{1}{2K_x f\left(1/z_{\min} - 1/z_{\max}\right)}. \tag{4.1}$$

K_x is the highest spatial frequency represented in the sampled light field, determined by the highest spatial frequency in the continuous light field B_x and the

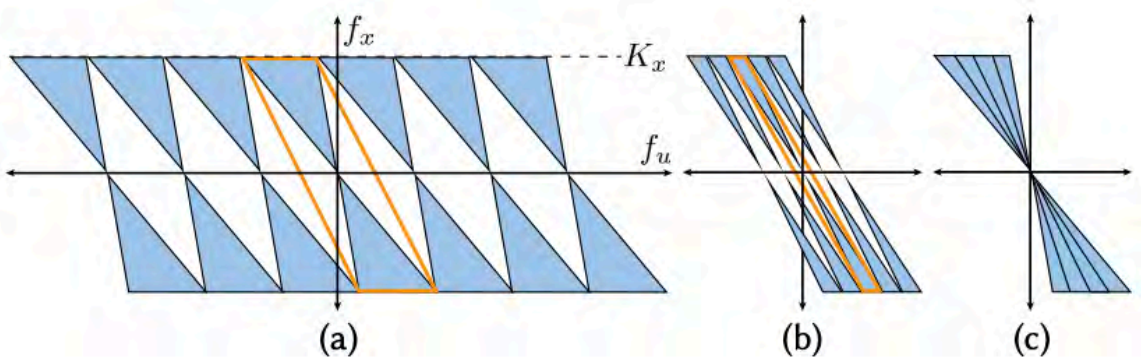


Figure 4.2. Traditional plenoptic sampling without occlusions, as derived in [9]. (a) The Fourier support of a light field without occlusions lies within a double-wedge, shown in blue. Nyquist rate view sampling is set by the double-wedge width, which is determined by the minimum and maximum scene depths $[z_{\min}, z_{\max}]$ and the maximum spatial frequency K_x. The ideal reconstruction filter is shown in orange. (b) Splitting the light field into D non-overlapping layers with equal disparity width decreases the Nyquist rate by a factor of D. (c) Without occlusions, the full light field spectrum is the sum of the spectra from each layer.

camera spatial resolution Δ_x:

$$K_x = \min\left(B_x, \frac{1}{2\Delta_x}\right). \tag{4.2}$$

4.3.2 MPI Scene Representation and Rendering

The MPI scene representation [166] consists of a set of fronto-parallel RGBα planes, evenly sampled in disparity within a reference camera's view frustum (see Figure 4.4). We can render novel views from an MPI at continuously-valued camera poses within a local neighborhood by alpha compositing the color along rays into the novel view camera using the "over" operator [105]. This rendering procedure is equivalent to reprojecting each MPI plane onto the sensor plane of the novel view camera and alpha compositing the MPI planes from back to front, as observed in early work on volume rendering [70]. An MPI can be considered as an encoding of a local light field, similar to layered light field displays [148, 149].

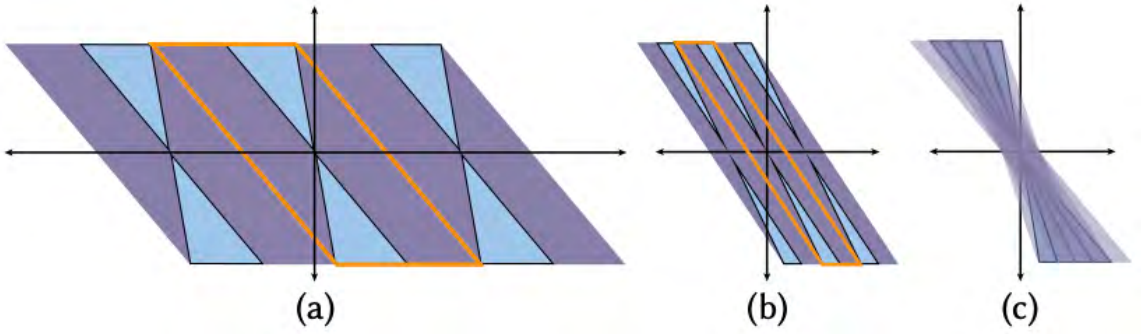


Figure 4.3. We extend traditional plenoptic sampling to consider occlusions when reconstructing a continuous light field from MPIs. (a) Considering occlusions expands the Fourier support to a parallelogram (the Fourier support without occlusions is shown in blue and occlusions expand the Fourier support to additionally include the purple region) and doubles the Nyquist view sampling rate. (b) As in the no-occlusions case, separately reconstructing the light field for D layers decreases the Nyquist rate by a factor of D. (c) With occlusions, the full light field spectrum cannot be reconstructed by summing the individual layer spectra because the union of their supports is smaller than the support of the full light field spectrum (a). Instead, we compute the full light field by alpha compositing the individual light field layers from back to front in the primal domain.

4.3.3 View Sampling Rate Reduction

Plenoptic sampling theory [9] additionally shows that decomposing a scene into D depth ranges and separately sampling the light field within each range allows the camera sampling interval to be increased by a factor of D. This is because the spectrum of the light field emitted by scene content within each depth range lies within a tighter double-wedge that can be packed D times more tightly than the full scene's double-wedge spectrum. Therefore, a tighter reconstruction filter with a different shear can be used for each depth range, as illustrated in Figure 4.2b. The reconstructed light field, ignoring occlusion effects, is simply the sum of the reconstructions of all layers, as shown in Figure 4.2c.

However, it is not straightforward to extend this analysis to handle occlusions, because the union of the Fourier spectra for all depth ranges has a smaller support than the original light field with occlusions, as visualized in Figure 4.3c. Instead, we observe that reconstructing a full scene light field from these depth range light fields while respecting occlusions would be much easier given corresponding per-view opacities, or shield fields [72], for each layer. We could then easily alpha

composite the depth range light fields from back to front to compute the full scene light field.

Each alpha compositing step increases the Fourier support by convolving the previously-accumulated light field's spectrum with the spectrum of the occluding depth layer. As is well known in signal processing, the convolution of two spectra has a Fourier bandwidth equal to the sum of the original spectra's bandwidths. Figure 4.3b illustrates that the width of the Fourier support parallelogram for each depth range light field, considering occlusions, is:

$$2K_x f \left(1/z_{\min} - 1/z_{\max}\right)/D, \tag{4.3}$$

so the resulting reconstructed light field of the full scene will enjoy the full Fourier support width.

We apply this analysis to our algorithm by interpreting the predicted MPI layers at each camera sampling location as view samples of scene content within non-overlapping depth ranges, and noting that applying the optimal reconstruction filter [9] for each depth range is equivalent to reprojecting and then blending pre-multiplied RGBα planes from neighboring MPIs. Our MPI layers differ from layered renderings considered in traditional plenoptic sampling because we predict opacities in addition to color for each layer, which allows us to correctly respect occlusions while compositing the depth layer light fields.

In summary, we extend the layered plenoptic sampling framework to correctly handle occlusions by taking advantage of our predicted opacities, and show that this still allows us to increase the required camera sampling interval by a factor of D:

$$\Delta_u \leq \frac{D}{2K_x f \left(1/z_{\min} - 1/z_{\max}\right)}. \tag{4.4}$$

Our framework further differs from classic layered plenoptic sampling in that each MPI is sampled within a reference camera view frustum with a finite field of view, instead of the infinite field of view assumed in prior analyses [9, 162]. In order for the MPI prediction procedure to succeed, every point within the scene's bounding volume should fall within the frustums of at least two neighboring sampled views. The required camera sampling interval Δ_u is then additionally bounded by:

$$\Delta_u \leq \frac{W \Delta_x z_{\min}}{2f} \tag{4.5}$$

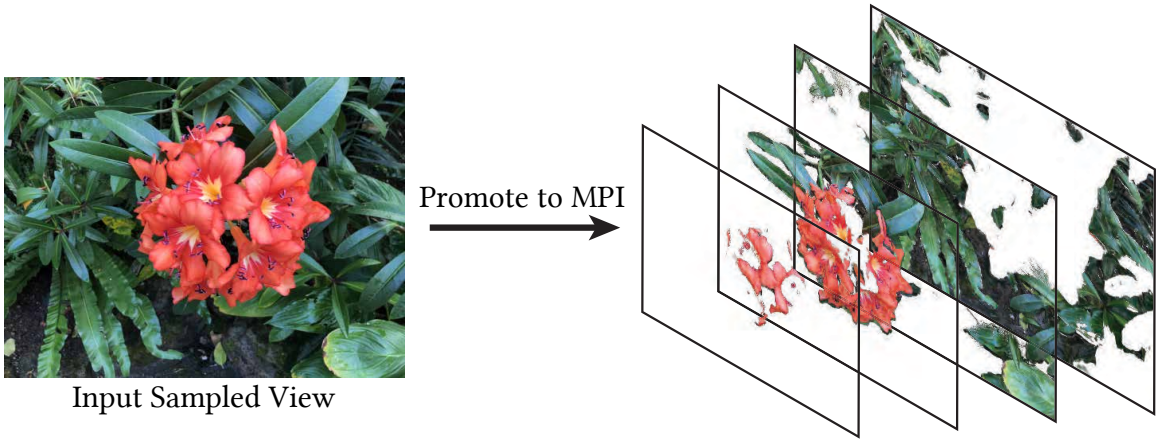


Figure 4.4. We promote each input view sample to an MPI scene representation [166], consisting of D RGBα planes at regularly sampled disparities within the input view's camera frustum. Each MPI can render continuously-valued novel views within a local neighborhood by alpha compositing color along rays into the novel view's camera.

where W is the image width in pixels of each sampled view. The overall camera sampling interval must satisfy both constraints:

$$\Delta_u \leq \min\left(\frac{D}{2K_x f\left(1/z_{\min} - 1/z_{\max}\right)}, \frac{W\Delta_x z_{\min}}{2f}\right). \tag{4.6}$$

4.3.4 Image Space Interpretation of View Sampling

It is useful to interpret the required camera sampling rate in terms of the maximum pixel disparity $d_{\max}$ of any scene point between adjacent input views. If we set $z_{\max} = \infty$ to allow scenes with content up to an infinite depth and additionally set $K_x = 1/2\Delta_x$ to allow spatial frequencies up to the maximum representable frequency:

$$\frac{\Delta_u f}{\Delta_x z_{\min}} = d_{\max} \leq \min\left(D, \frac{W}{2}\right). \tag{4.7}$$

Simply put, the maximum disparity of the closest scene point between adjacent views must be less than $\min(D, W/2)$ pixels. When $D = 1$, this inequality reduces to the Nyquist bound: a maximum of 1 pixel of disparity between views.

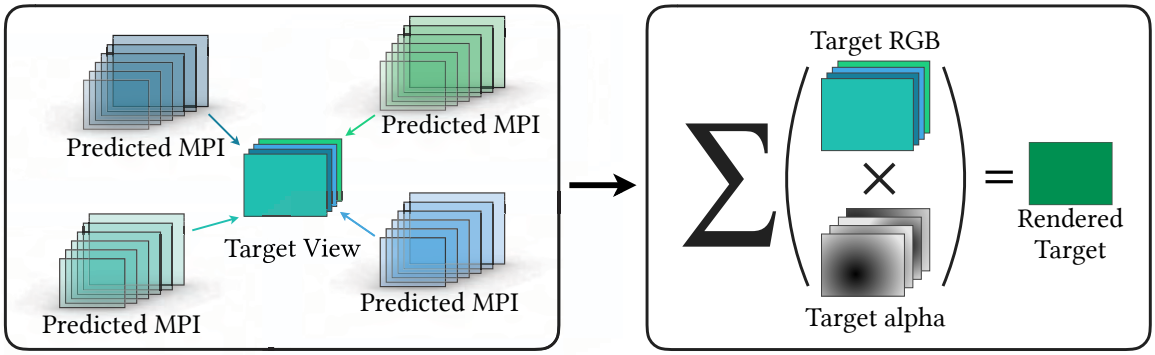


Figure 4.5. We render novel views as a weighted combination of renderings from neighboring MPIs, modulated by the corresponding accumulated alphas.

In summary, promoting each view sample to an MPI scene representation with D depth layers allows us to decrease the required view sampling rate by a factor of D, up to the required field of view overlap for stereo geometry estimation. Light fields for real 3D scenes must be sampled in two viewing directions, so this benefit is compounded into a sampling reduction of D^2. Section 4.6.1 empirically validates that our algorithm's performance matches this theoretical analysis. Section 4.7.1 describes how we apply the above theory along with the empirical performance of our deep learning pipeline to prescribe practical sampling guidelines for users.

4.4 Practical View Synthesis Pipeline

We present a practical and robust method for synthesizing new views from a set of input images and their camera poses. Our method first uses a CNN to promote each captured input image to an MPI, then reconstructs novel views by blending renderings from nearby MPIs. Figures 4.1 and 4.5 visualize this pipeline. We discuss the practical image capture process enabled by our method in Section 4.7.

4.4.1 MPI Prediction for Local Light Field Expansion

The first step in our pipeline is expanding each sampled view to a local light field using an MPI scene representation. Our MPI prediction pipeline takes five views as input: the reference view to be expanded and its four nearest neighbors in 3D space. Each image is reprojected to D depth planes, sampled linearly in disparity within the reference view frustum, to form 5 plane sweep volumes (PSVs) of size $H \times W \times D \times 3$.

Our 3D CNN takes these 5 PSVs as input, concatenated along the channel dimension. This CNN outputs an opacity α for each MPI coordinate (x, y, d) as well as a set of 5 color selection weights that sum to 1 at each MPI coordinate. These weights parameterize the RGB values in the output MPI as a weighted combination of the input PSVs. Intuitively, each predicted MPI softly "selects" its color values at each MPI coordinate from the pixel colors at that coordinate in each of the input PSVs. We specifically use this RGB parameterization instead of the foreground+background parameterization proposed by Zhou *et al*. [166] because their method does not allow an MPI to directly incorporate content occluded from the reference view but visible in other input views.

Furthermore, we enhance the MPI prediction CNN architecture from the original version to use 3D convolutional layers instead of the original 2D convolutional layers so that our architecture is fully convolutional along the height, width, and depth dimensions. This enables us to predict MPIs with a variable number of planes D so that we can jointly choose the view and disparity sampling densities to satisfy Equation 4.7. Table 4.2 validates the benefit of being able to change the number of MPI planes to correctly match our derived sampling requirements, enabled by our use of 3D convolutions.

4.4.2 Continuous View Reconstruction by Blending

As discussed in Section 4.3, we reconstruct interpolated views as a weighted combination of renderings from multiple nearby MPIs. This effectively combines our local light field approximations into a light field with a near plane spanning the extent of the captured input views and a far plane determined by the field-of-view of the input views. As in standard light field rendering, this allows for a new view path with unconstrained 3D translation and rotation within the range of views made up of rays in the light field.

One important detail in our rendering process is that we consider the accumu-

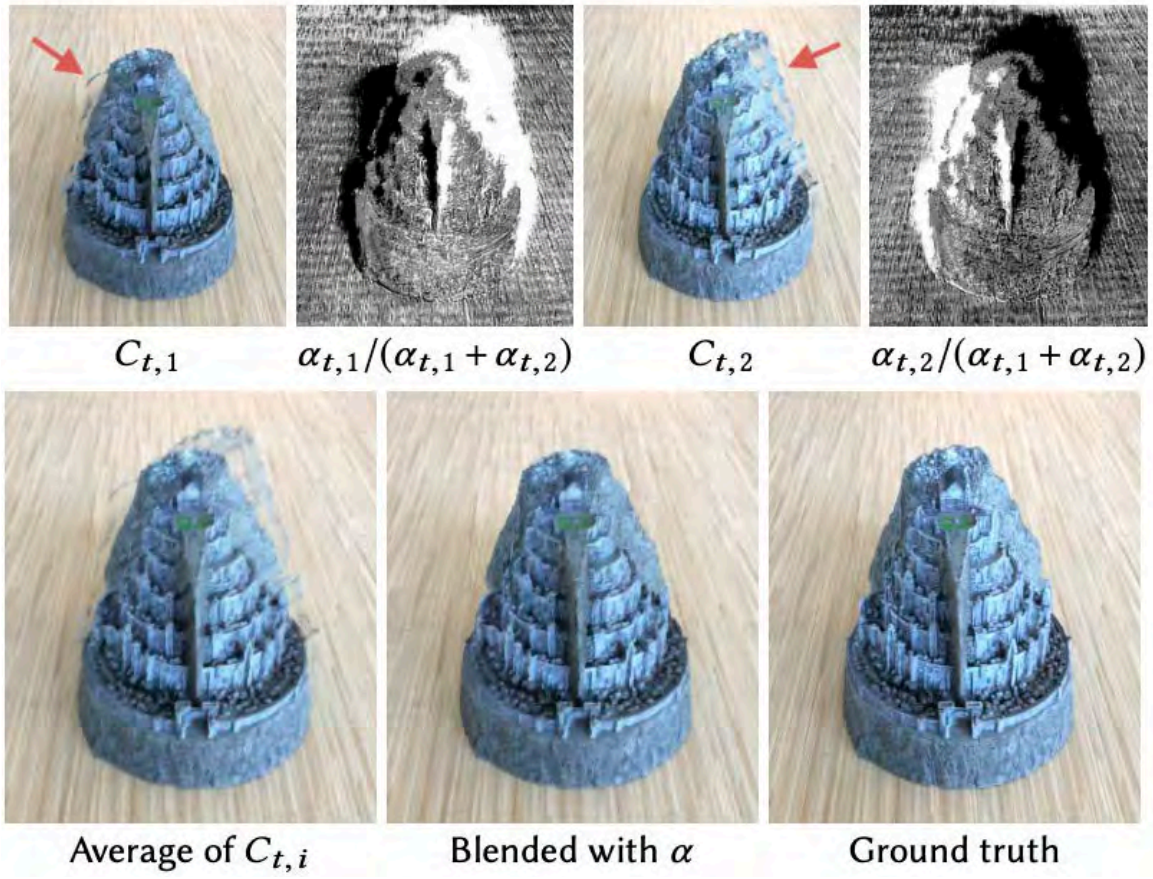


Figure 4.6. An example illustrating the benefits of using accumulated alpha to blend MPI renderings. We render two MPIs at the same new camera pose. In the top row, we display the RGB outputs $C_{t,i}$ from each MPI as well as the accumulated alphas $\alpha_{t,i}$, normalized so that they sum to one at each pixel. In the bottom row, we see that a simple average of the RGB images $C_{t,i}$ retains the stretching artifacts from both MPI renderings, whereas the alpha weighted blending combines only the non-occluded pixels from each input to produce a clean output C_t.

lated alpha values from each MPI rendering when blending. This allows each MPI rendering to "fill in" content that is occluded from other camera views.

Our MPI prediction network uses a set of RGB images C_k along with their camera poses p_k to produce a set of MPIs M_k (one corresponding to each input image). To render a novel view with pose p_t using the predicted MPI M_k, we homography warp each RGBα MPI plane into the frame of reference of the target pose p_t then alpha composite the warped planes together from back to front. This produces an RGB image and an alpha image, which we denote $C_{t,k}$ and $\alpha_{t,k}$ respectively (subscript t, k indicating that the output is rendered at pose p_t using the MPI at pose p_k).

Since a single MPI alone will not necessarily contain all the content visible from the new camera pose due to occlusions and field of view issues, we generate the final RGB output C_t by blending rendered RGB images $C_{t,k}$ from multiple MPIs, as depicted in Figure 4.5. We use scalar blending weights $w_{t,k}$, each modulated by the corresponding accumulated alpha images $\alpha_{t,k}$ and normalized so that the resulting rendered image is fully opaque ($\alpha = 1$):

$$C_t = \frac{\sum_k w_{t,k}\alpha_{t,k}C_{t,k}}{\sum_k w_{t,k}\alpha_{t,k}}. \tag{4.8}$$

For an example where modulating the blending weights by the accumulated alpha values prevents artifacts in C_t, see Figure 4.6. Table 4.2 demonstrates that blending with alpha gives quantitatively superior results over both using a single MPI and blending multiple MPI renderings without using the accumulated alpha.

The blending weights $w_{t,k}$ can be any sufficiently smooth filter. In the case of data sampled on a regular grid, we use bilinear interpolation from the four nearest MPIs rather than the ideal sinc function interpolation for effiency and due to the limited number of sampled views. For irregularly sampled data, we use the five nearest MPIs and take $w_{t,k} \propto \exp\left(-\gamma\ell(p_t, p_k)\right)$. Here $\ell(p_t, p_k)$ is the L^2 distance between the translation vectors of poses p_t and p_k, and the constant γ is defined as $\frac{f}{Dz_{\min}}$ given focal length f, minimum distance to the scene $z_{\min}$, and number of planes D. (Note that the quantity $\frac{f\ell}{z_{\min}}$ represents ℓ converted into units of pixel disparity.)

Our strategy of blending between neighboring MPIs is particularly effective for rendering non-Lambertian effects. For general curved surfaces, the virtual apparent depth of a specularity changes with the viewpoint [135]. As a result, specularities appear as curves in epipolar slices of the light field, while diffuse points appear as lines. Each of our predicted MPIs can represent a specularity for a local range of

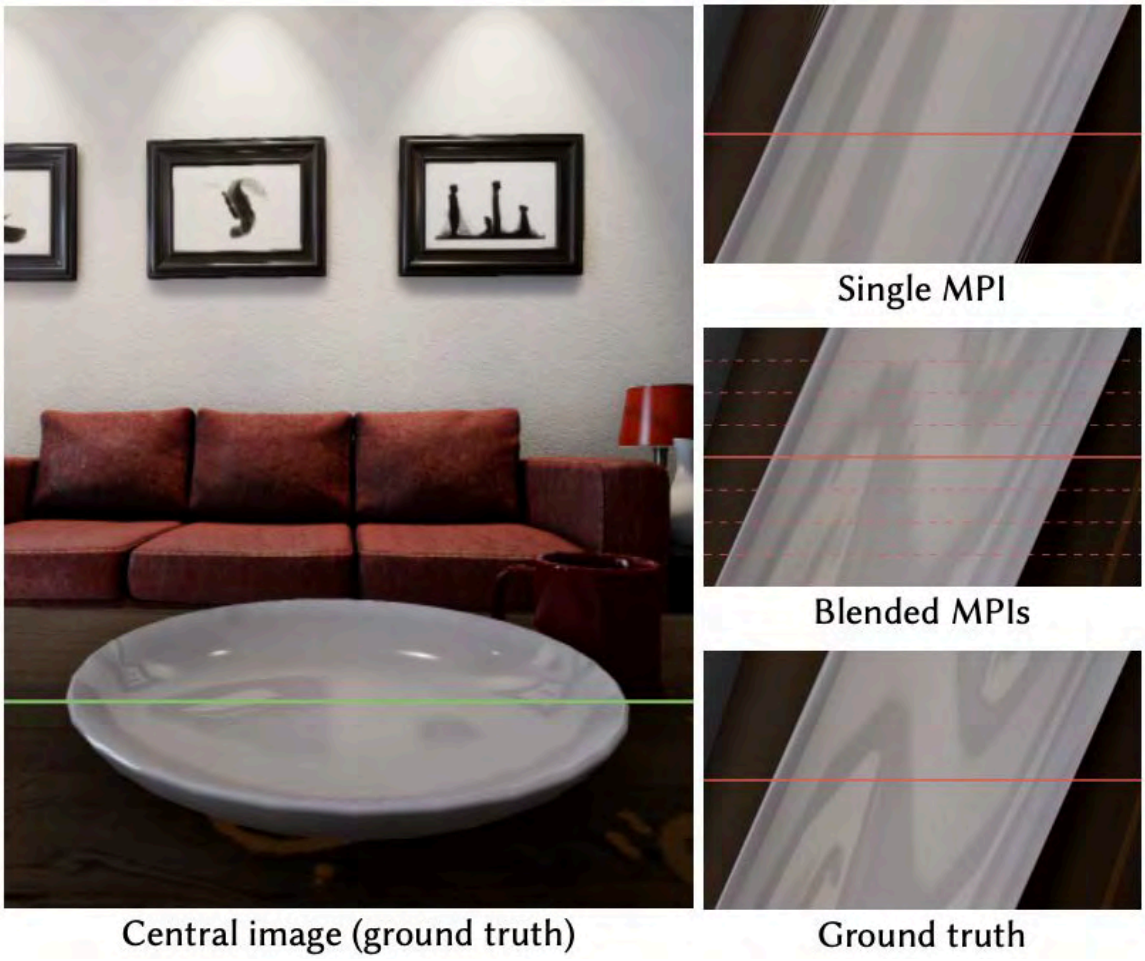


Figure 4.7. We demonstrate that a collection of MPIs can approximate a highly non-Lambertian light field. In this synthetic scene, the curved plate reflects the paintings on the wall, leading to quickly-varying specularities as the camera moves horizontally. This effect can be observed in the ground truth epipolar plot (bottom right). A single MPI (top right) can only place a specular reflection at a single virtual depth, but blending renderings from multiple MPIs (middle right) provides a much better approximation to the true light field. In this example, we blend between MPIs evenly distributed at every 32 pixels of disparity along a horizontal path, indicated by the dashed lines in the epipolar plot.

views by placing the specularity at a single virtual depth. Figure 4.7 illustrates how our rendering procedure effectively models a specularity's curve in the light field by blending locally linear approximations, as opposed to the limited extrapolation provided by a single MPI.

4.5 Training Our View Synthesis Pipeline

4.5.1 Training Dataset

We train our view synthesis pipeline using both renderings and real images of natural scenes. Using synthetic training data crucially enables us to easily generate a large dataset with input view and scene depth distributions similar to those we expect at test time, while using real data helps us generalize to real-world lighting and reflectance effects as well as small errors in pose estimation.

Our synthetic training set consists of images rendered from the SUNCG [128] and UnrealCV [106] datasets. SUNCG contains 45,000 simplistic house and room environments with texture mapped surfaces and low geometric complexity. UnrealCV contains only a few large scale environments, but they are modeled and rendered with extreme detail, providing geometric complexity, texture variety, and non-Lambertian reflectance effects. We generate views for each synthetic training instance by first randomly sampling a target baseline for the inputs (up to 128 pixels of disparity), then randomly perturbing the camera pose in 3D to approximately match this baseline.

Our real training dataset consists of 24 scenes from our handheld cellphone captures, with 20-30 images each. We use the COLMAP structure from motion [113] implementation to compute poses for our real images.

4.5.2 Training Procedure

For each training step, we sample two sets of 5 views each to use as inputs, and a single held-out target view for supervision. We first use the MPI prediction network to predict two MPIs, one from each set of 5 inputs. Next, we render the target novel view from both MPIs and blend these renderings using the accumulated alpha values, as described in Equation 4.8.

The training loss is simply the image reconstruction loss for the rendered novel view. We follow the original work on MPI prediction [166] and use a VGG network activation perceptual loss as implemented by Chen and Koltun [12], which has been consistently shown to outperform standard image reconstruction losses [55, 163]. We are able to supervise only the final blended rendering because both our fixed rendering and blending functions are differentiable. Learning through this blending step trains our MPI prediction network to leave alpha "holes" in uncertain regions for each MPI, in the expectation that this content will be correctly rendered by another neighboring MPI, as illustrated by Figure 4.6.

In practice, training through blending is slower than training a single MPI, so we first train the network to render a new view from only one MPI for 500k iterations, then train the full pipeline (blending views from two different MPIs) for 100k iterations. To fine tune the network to process real data, we train on our small real dataset for an additional 10k iterations. We use 320×240 resolution and up to 128 planes for SUNCG training data, and 640×480 resolution and up to 32 planes for UnrealCV training data, due to GPU memory limitations. We implement our full pipeline in Tensorflow [1] and optimize the MPI prediction network parameters using Adam [67] with a learning rate of 2×10^{-4} and a batch size of one. We split the training pipeline across two Nvidia RTX 2080Ti GPUs, using one GPU to generate each MPI.

4.6 Experimental Evaluation

We quantitatively and qualitatively validate our method's prescriptive sampling benefits and ability to render high fidelity novel views of light fields that have been undersampled by up to $4000\times$, as well as demonstrate that our algorithm outperforms state-of-the-art methods for regular view interpolation. Figure 4.9 showcases these qualitative comparisons on scenes with complex geometry (Fern and T-Rex) and highly non-Lambertian scenes (Air Plants and Pond) that are not handled well by most view synthesis algorithms.

For all quantitative comparisons (Table 4.2), we use a synthetic test set rendered from an UnrealCV [106] environment that was not used to generate any training data. Our test set contains 8 scenes, each rendered at 640×480 resolution and at 8 different view sampling densities such that the maximum disparity between adjacent input views ranges from 1 to 256 pixels (a maximum disparity of 1 pixel between input views corresponds to Nyquist rate view sampling). We restrict our

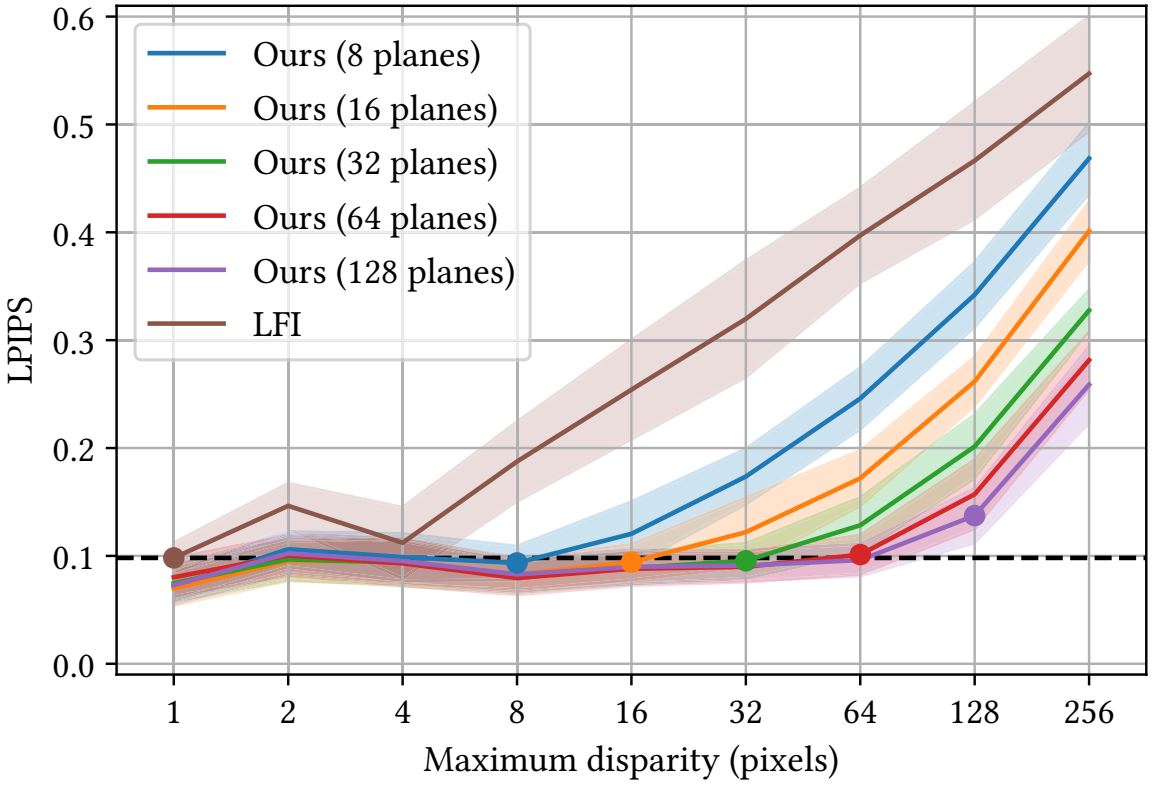


Figure 4.8. We plot the performance of our method (with varying number of planes $D = 8, 16, 32, 64$, and 128) compared to light field interpolation for different input view sampling rates (denoted by maximum scene disparity $d_{\max}$ between adjacent input views). Our method can achieve the same perceptual quality as LFI with Nyquist rate sampling (black dotted line) as long as the number of predicted planes matches or exceeds the undersampling rate, up to an undersampling rate of 128. At $D = 64$, this means we achieve the same quality as LFI with $64^2 \approx 4000\times$ fewer views. We use the LPIPS [163] metric (lower is better) because we primarily value perceptual quality. The colored dots indicate the point on each line where the number of planes equals the maximum scene disparity, where equality is achieved in our sampling bound (Equation 4.7). The shaded region indicates ± 1 standard deviation over all 8 test scenes.

quantitative comparisons to rendered images because a Nyquist rate grid-sampled light field would require at least 384^2 camera views to generate a similar test set, and no such densely-sampled real light field dataset exists to the best of our knowledge. We report quantitative performance using the standard PSNR and SSIM metrics, as well as the state-of-the-art LPIPS [163] perceptual metric, which is based on a weighted combination of neural network activations tuned to match human judgements of image similarity.

Finally, our accompanying video shows results on over 60 additional real-world scenes. These renderings were created completely automatically by a script that takes only the set of captured images and desired output view path as inputs, highlighting the practicality and robustness of our method.

4.6.1 Sampling Theory Validation

Our method is able to render high-quality novel views while significantly decreasing the required input view sampling density compared to standard light field interpolation. Figure 4.8 shows that our method is able to render novel views with Nyquist level perceptual quality with up to $d_{\max} = 64$ pixels of disparity between input view samples, as long as we match the number of planes in each MPI to the maximum pixel disparity between input views. We postulate that our inability to match Nyquist quality from input images with a maximum of 128 pixels of disparity is due to the effect of occlusions. It becomes increasingly likely that any non-foreground scene point will be sampled by fewer input views as the maximum disparity between adjacent views increases. This increases the difficulty of depth estimation and requires the CNN to hallucinate the appearance and depth of occluded points in extreme cases where they are sampled by none of the input views.

Figure 4.8 also shows that once our sampling bound is satisfied, adding additional planes does not increase performance. For example, at 32 pixels of disparity, increasing from 8 to 16 to 32 planes decreases the LPIPS error, but performance stays constant from 32 to 128 planes. This verifies that for scenes up to 64 pixels of disparity, adding additional planes past the maximum pixel disparity between input views is of limited value, in accordance with our theoretical claim that partitioning a scene with disparity variation of D pixels into D depth ranges is sufficient for continuous reconstruction.

Table 4.2. We quantitatively show that our method outperforms state-of-the-art baselines and specific ablations of our method, across a wide range of input sampling rates (measured by the maximum pixel disparity $d_{\max}$ between adjacent input views), on a synthetic test set. We display results using the standard PSNR and SSIM metrics (higher is better) as well as the LPIPS perceptual metric [163] (lower is better). The best measurement in each column is bolded. See Sections 4.6.2 and 4.6.3 for details on each comparison.

		Maximum disparity $d_{\max}$ (pixels)											
		16			32			64			128		
	Algorithm	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Baselines	LFI	26.21	0.7776	0.2541	23.35	0.6982	0.3198	20.60	0.6243	0.3971	18.32	0.5560	0.4665
	ULR	28.17	0.8320	0.1510	26.43	0.7987	0.1820	24.34	0.7679	0.2311	21.24	0.7062	0.3215
	Soft3D	34.48	0.9430	0.1345	32.33	0.9216	0.1795	27.97	0.8588	0.2652	23.11	0.7382	0.3979
	BW Deep	34.18	0.9433	0.1074	34.00	0.9476	0.1128	31.88	0.9192	0.1573	27.59	0.8363	0.2591
Ablations	Single MPI	31.11	0.9482	0.1007	29.38	0.9424	0.1111	26.88	0.9250	0.1363	24.20	0.8734	0.1980
	Avg. MPIs	32.67	0.9560	0.1140	31.34	0.9532	0.1248	29.31	0.9400	0.1423	27.02	0.8999	0.1961
	Ours	**34.57**	**0.9568**	**0.0942**	**34.48**	**0.9569**	**0.0954**	**33.58**	**0.9530**	**0.1012**	**31.96**	**0.9323**	**0.1374**

4.6.2 Comparisons to Baseline Methods

We quantitatively (Table 4.2) and qualitatively (Figure 4.9) demonstrate that our algorithm produces superior renderings, particularly for non-Lambertian effects, without the artifacts seen in renderings from competing methods. We urge readers to view our accompanying video for convincing rendered camera paths that highlight the benefits of our approach.

We compare our method to state-of-the-art view synthesis techniques as well as view-dependent texture mapping using a global mesh as proxy geometry.

Light Field Interpolation (LFI) [9] This baseline is representative of continuous view reconstruction based on classic signal processing. Following the method of plenoptic sampling [9], we render novel views using a bilinear interpolation reconstruction filter sheared to the mean scene disparity. Figure 4.9 demonstrates that increasing the camera spacing beyond the Nyquist rate results in aliasing and ghosting artifacts when using this method.

Unstructured Lumigraph Rendering (ULR) [7] This baseline is representative of view dependent texture mapping with an estimated global mesh as a geometry proxy. We reconstruct a global mesh from all inputs using the screened Poisson surface reconstruction algorithm [64], and use the heuristic Unstructured Lumigraph blending weights [7] to blend input images after reprojecting them into the novel viewpoint using the global mesh. We use a plane at the mean scene disparity as a proxy geometry to fill in holes in the mesh.

It is particularly difficult to reconstruct a global mesh with geometry edges that are well-aligned with image edges, which causes perceptually jarring artifacts. Furthermore, mesh reconstruction often fails to fill in large portions of the scene, resulting in ghosting artifacts similar to those seen in light field interpolation.

Soft3D [103] Soft3D is a state-of-the-art view synthesis algorithm that is similar to our approach in that it also computes a local layered scene representation for each input view and projects and blends these volumes to render each novel view. However, it uses a hand-crafted pipeline based on classic local stereo and guided filtering to compute each layered representation. Furthermore, since classic stereo methods are unreliable for smooth or repetitive image textures and non-Lambertian materials, Soft3D relies on smoothing their geometry estimation across many (up to 25) input views.

Table 4.2 quantitatively demonstrates that our approach outperforms Soft3D overall. In particular, Soft3D's performance degrades much more rapidly as the input view sampling rate decreases since their aggregation is less effective when fewer input images view the same scene content. Our method is able to predict high-quality geometry in scenarios where Soft3D suffers from noisy and erroneous results of local stereo because we leverage deep learning to learn implicit priors on natural scene geometry. This is in line with recent work that has shown the benefits of deep learning over traditional stereo for depth estimation [66, 55].

Figure 4.9 qualitatively demonstrates that Soft3D generally contains blurred geometry artifacts due to errors in local depth estimation, and that Soft3D's approach fails for rendering non-Lambertian effects because their aggregation procedure blurs the specularity geometry, which changes with the input image viewpoint.

Backwards warping deep network (BW Deep) This baseline subsumes recent deep learning view synthesis techniques [61, 31], which use a CNN to estimate geometry for each novel view and then backwards warp and blend nearby input

images to render the target view. We train a network that uses the same 3D CNN architecture as our MPI prediction network but instead outputs a single depth map at the pose of the new target view. We then backwards warp the five input images into the new view using this depth map and use a second 2D CNN to composite these warped input images into a single rendered output view. As shown in Table 4.2, performance for this method degrades quickly as the maximum disparity increases. Although this approach produces comparable images to our method for scenes with small disparities ($d_{\max} = 16, 32$), the renderings suffer from extreme inconsistency when rendering video sequences.

BW Deep methods use a CNN to estimate depth separately for each output viewpoint, so artifacts appear and disappear over only a few frames, resulting in rapid flickers and pops in the rendered camera path. This inconsistency is visible as corruption in the epipolar plots in Figure 4.9. Furthermore, backwards warping incentivizes incorrect depth predictions to fill in disocclusions, so BW Deep methods also produce errors around thin structures and occlusion edges.

4.6.3 Ablation Studies

We validate our overall strategy of blending between multiple MPIs as well as our specific blending procedure using accumulated alphas with the following ablation studies:

Single MPI The fifth row of Table 4.2 shows that using only one MPI to produce new views results in significantly decreased performance due to the limited field of view represented in a single MPI as well as depth discretization artifacts as the target view moves far from the MPI reference viewpoint. Additionally, Figure 4.7 shows an example of complex non-Lambertian reflectance that cannot be represented by a single MPI. This ablation can be considered an upper bound on the performance of Zhou *et al*. [166], since we use one MPI generated by a higher capacity 3D CNN.

Average MPIs The sixth row of Table 4.2 shows that blending multiple MPI outputs for each novel view without using the accumulated alpha channels results in decreased performance. Figure 4.6 visualizes that this simple blending leads to ghosting in regions that are occluded from the poses of any of the MPIs used for rendering, because they will contain incorrect content in disoccluded regions.

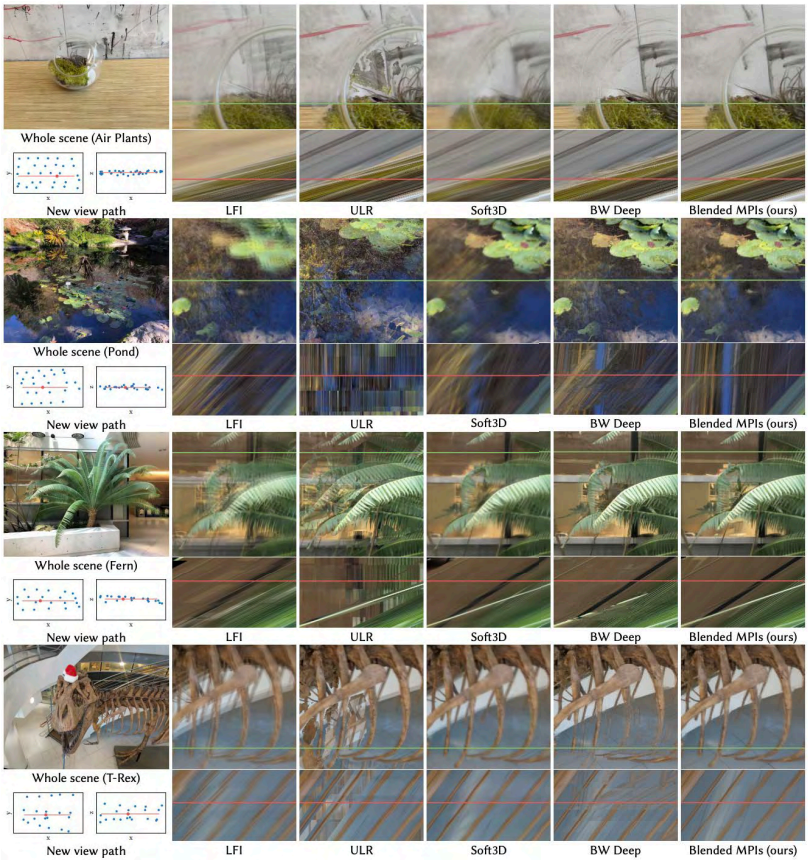


Figure 4.9. Results on real cellphone datasets. We render a sequence of new views and show both a crop from a single rendered output and an epipolar slice of the sequence. We show 2D projections of the input camera poses (blue dots) and new view path (red line) along the z and y axes of the new view camera in the lower left of each row. LFI fails to cleanly represent objects at different depths because it only uses a single depth plane for reprojection, leading to ghosting (leaves in Fern, lily pads in Pond) and depth inconsistency visible in all epipolar images. Mesh reconstruction failures cause artifacts visible in both the crops and epipolar images for ULR. Soft3D's depth uncertainty leads to blur, and geometry aggregation across large view neighborhoods results in incorrect specularity geometry (brown and blue reflections in Pond). BW Deep's use of a CNN to render every novel view causes depth inconsistency, visible as choppiness across the rows of the epipolar images in all examples. Additionally, BW Deep selects a single depth per pixel, leading to errors for transparencies (glass rim in Air Plants) and reflections (Pond). BW Deep also uses backwards warping, which causes errors around occlusion boundaries (thin ribs in T-Rex).

4.7 Practical Usage

We present guidelines to assist users in sampling views that enable high-quality view interpolation with our algorithm, and showcase our method's practicality with a smartphone camera app that guides users to easily capture such input images. Furthermore, we implement a mobile viewer that renders novel views from our predicted MPIs in real-time. Figure 4.9 showcases examples of rendered results from handheld smartphone captures. Our accompanying video contains a screen capture of our app in use, as well as results on over 60 real-world scenes generated by an automated script.

4.7.1 Prescriptive Scene Sampling Guidelines

In a typical capture scenario, a user will have a camera with a field of view θ and a world space plane with side length S that bounds the viewpoints they wish to render. Based on this, we prescribe the design space of image resolution W and number of images to sample N that users can select from to reliably render novel views at Nyquist-level perceptual quality.

Section 4.6.1 shows that the empirical limit on the maximum disparity $d_{\max}$ between adjacent input views for our deep learning pipeline is 64 pixels. Substituting Equation 4.7:

$$\frac{\Delta_u f}{\Delta_x z_{\min}} \leq 64. \tag{4.9}$$

We translate this into user-friendly quantities by noting that $\Delta_u = S/\sqrt{N}$ and that the ratio of sensor width to focal length $W\Delta_x/f = 2\tan\theta/2$:

$$\frac{W}{\sqrt{N}} \leq \frac{128 z_{\min} \tan(\theta/2)}{S}. \tag{4.10}$$

Using a smartphone camera with a 64° field of view, this is simply:

$$\frac{W}{\sqrt{N}} \leq \frac{80 z_{\min}}{S}. \tag{4.11}$$

Intuitively, once a user has determined the extent of viewpoints they wish to render and the depth of the closest scene point, they can choose any target rendering resolution W and number of images to capture N such that the ratio $W/\sqrt{N}$ satisfies the above expression.

4.7.2 Asymptotic Rendering Time and Space Complexity

Within the possible choices of rendering resolution W and number of sampled views N that satisfy the above guideline, different users may value capture time, rendering time, and storage costs differently. We derive the asymptotic complexities of these quantities to further assist users in choosing correct parameters for their application.

First, the capture time is simply $O(N)$. The render time of each MPI generated is proportional to the number of planes times the pixels per plane:

$$W^2 D = \frac{W^3 S}{2\sqrt{N} z_{\min} \tan(\theta/2)} = O(W^3 N^{-1/2}). \tag{4.12}$$

Note that the rendering time for each MPI decreases as the number of sampled images N increases, because this allows us to use fewer planes per MPI. The total MPI storage cost is proportional to:

$$W^2 D \cdot N = \frac{W^3 S \sqrt{N}}{2 z_{\min} \tan(\theta/2)} = O(W^3 N^{1/2}). \tag{4.13}$$

Practically, this means that users should determine their specific rendering time and storage constraints, and then maximize the image resolution and number of sampled views that satisfy their constraints as well as the guideline in Equation 4.10. Figure 4.10 visualizes these constraints for an example user.

4.7.3 Smartphone Capture App

We develop an app for iOS smartphones, based on the ARKit framework, that guides users to capture input views for our view synthesis algorithm. The user first taps the screen to mark the closest object, and the app uses the corresponding scene depth computed by ARKit as $z_{\min}$. Next, the user selects the size of the view

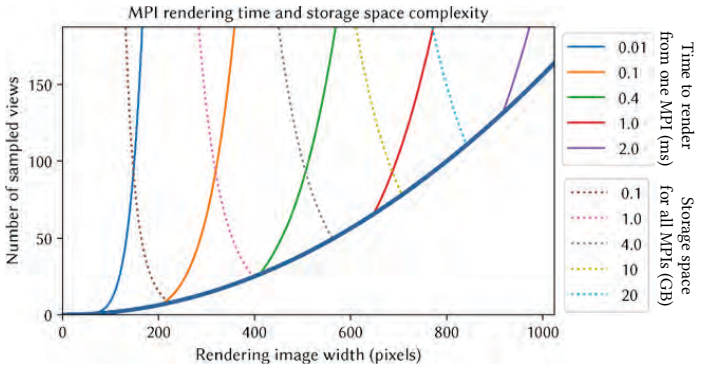


Figure 4.10. Time and storage cost tradeoff within the space of rendering resolution and number of sampled views that result in Nyquist level perceptual quality (space above the thick blue curve signifying $D = d_{\max} \leq 64$, as in Equation 4.11). We plot isocontours of rendering time and storage space for an example scene with close depth $z_{\min} = 1.0m$ and target view plane with side length $0.5m$, captured with a camera with a 64° field of view. We use the average rendering speed from our desktop viewer and the storage requirement from uncompressed 8-bit MPIs. Users can select the point where their desired rendering speed and storage space isocontours intersect to determine the minimum required number of views and maximum affordable rendering resolution.

(a) Grid of guides shows user where to move phone

(b) Image automatically captured when phone aligns with guide

Figure 4.11. Equation 4.7 prescribes a simple sampling bound related only to the maximum scene disparity. We take advantage of the augmented reality toolkits available in modern smartphones to create an app that helps the user sample a real scene for rendering with our method. (a) We use built-in software to track the phone's position and orientation, providing sampling guides that allow the user to space photos evenly at the target disparity. (b) Once the user has centered the phone so that the RGB axes align with one of the guides, the app automatically captures a photo.

plane S within which our algorithm will render novel views. We fix the rendering resolution for the smartphone app to $W = 500$ which therefore fixes the prescribed number and spacing of required images based on Equation 4.11 and the definition $\Delta_u = S/\sqrt{N}$. Our app then guides the user to capture these views using the intuitive augmented reality overlay shown in Figure 4.11. When the phone detects that the camera has been moved to a new sample location, it automatically records an image and highlights the next sampling point.

4.7.4 Preprocessing

After capturing the required input images, the only preprocessing required before being able to render novel views is estimating the input camera poses and using our trained network to predict an MPI for each input view. Unfortunately, camera poses from ARKit are currently not accurate enough for acceptable results, so we use the open source COLMAP software package [113, 114], which takes about 2-6 minutes for sets of 20-30 input images.

We use the deep learning pipeline described in Section 4.4.1 to predict an MPI for each input sampled view. On an Nvidia GTX 1080Ti GPU, This takes approximately 0.5 seconds for a small MPI ($500 \times 350 \times 32 \approx 6$ megavoxels) or 12 seconds for a larger MPI that must be output in overlapping patches ($1000 \times 700 \times 64 \approx 45$ megavoxels). In total, our method only requires about 10 minutes of preprocessing to estimate poses and predict MPIs before being able to render novel views at a 1 megapixel image resolution.

With the increasing investment in smartphone AR and on-device deep learning accelerators, we expect that smartphone pose estimation will soon be accurate enough and on-device network inference will be powerful enough for users to go from capturing images to rendering novel views within a few seconds.

4.7.5 Real-Time Viewers

We implement novel view rendering from a single MPI by rasterizing each plane from back to front using texture mapped rectangles in 3D space, invoking a standard shader API to correctly handle the alpha compositing, perspective projection, and texture resampling. For each new view, we determine the MPIs to be blended, as discussed in Section 4.4.2, and render them into separate framebuffers. We then

use a simple fragment shader to perform the alpha-weighted blending described in Section 4.4.2. We implement this rendering pipeline as desktop viewer using OpenGL which renders views with 1000×700 resolution at 60 frames per second, as well as an iOS mobile viewer using the Metal API which renders views with 500×350 resolution at 30 frames per second. Please see our video for demonstrations of these real-time rendering implementations.

4.7.6 Limitations

A main limitation of our algorithm is that the MPI network sometimes assigns high opacity to incorrect layers in regions of ambiguous or repetitive texture and regions where scene content moves between input images. This can cause floating or blurred patches in the rendered output sequence (see the far right side of the fern in our video), which is a common failure mode in methods that rely on texture matching cues to infer depth. These artifacts could potentially be ameliorated by using more input views to disambiguate stereo matching and by encouraging the network to learn stronger global priors on 3D geometry.

Another limitation is the difficulty of scaling to higher image resolutions. As evident in Equations 4.12 and 4.13, layered approaches such as our method are limited by complexities that scale cubically with the image width in pixels. Furthermore, increasing the image resolution requires a CNN with a larger receptive field. This could be addressed by exploring multiresolution CNN architectures and hierarchical volume representations such as octrees, or by predicting a more compact local scene representation such as layered depth images [117] with opacity.

4.8 Discussion

We have presented a simple and practical method for view synthesis that works reliably for complex real-world scenes, including non-Lambertian materials. Our algorithm first promotes each input image into a layered local light field representation, then renders novel views in real time by blending outputs generated by nearby representations. We extend traditional layered plenoptic sampling analysis to handle occlusions and provide a theoretical sampling bound on how many views are needed for our method to render high-fidelity views of a given scene. We quantitatively validate this bound and demonstrate that we match the perceptual quality of

dense Nyquist rate view sampling while using $\approx 4000\times$ fewer input images. Our accompanying video demonstrates that we thoroughly outperform prior work, and showcases results on over 60 diverse and complex real-world scenes, where our novel views are rendered with a fully automated capture-to-render pipeline. We believe that our work paves the way for future advances in image-based rendering that combine the empirical performance benefits of data-driven machine learning methods with the robust reliability guarantees of traditional geometric and signal processing based analysis.

In this chapter and the previous chapter, we have shown how multiplane images are an effective and parameter-efficient 3D volumetric representation of scenes for view synthesis. However, it is important to note that the multiplane image representation is specifically designed for rendering "forward-facing" novel views without significant camera rotation or zoom. In the next chapter, we present a different parameter-efficient volumetric scene representation for a view synthesis task where we are specifically interested in synthesizing novel views from locations *within* the scene and far away from the input viewpoints.

Chapter 5

Panoramic View Synthesis with Multiscale Volumes

In this chapter, we present a volumetric 3D scene representation for the task of estimating the global illumination seen at any point within a scene given images of the scene. Although illumination estimation is not typically treated as a view synthesis problem, it is indeed a special case of view synthesis with the goal of rendering a panoramic image, representing the incoming light, at any point within the scene. However, the relative positions of input and output viewpoints is quite different than the typical view synthesis setup: in illumination estimation, we typically have standard perspective view(s) of the scene from an "outside" viewpoint, and we would like to render a panoramic image within the "interior" of the scene (Figure 5.2) for the purpose of relighting an object inserted into the scene at that location. The multiplane image representation discussed earlier is not an adequate representation for synthesizing views within the scene, so in this chapter we present a multiscale volumetric representation that is particularly well-suited for the task of illumination estimation.

This chapter is based on joint work published at CVPR 2020 [131].

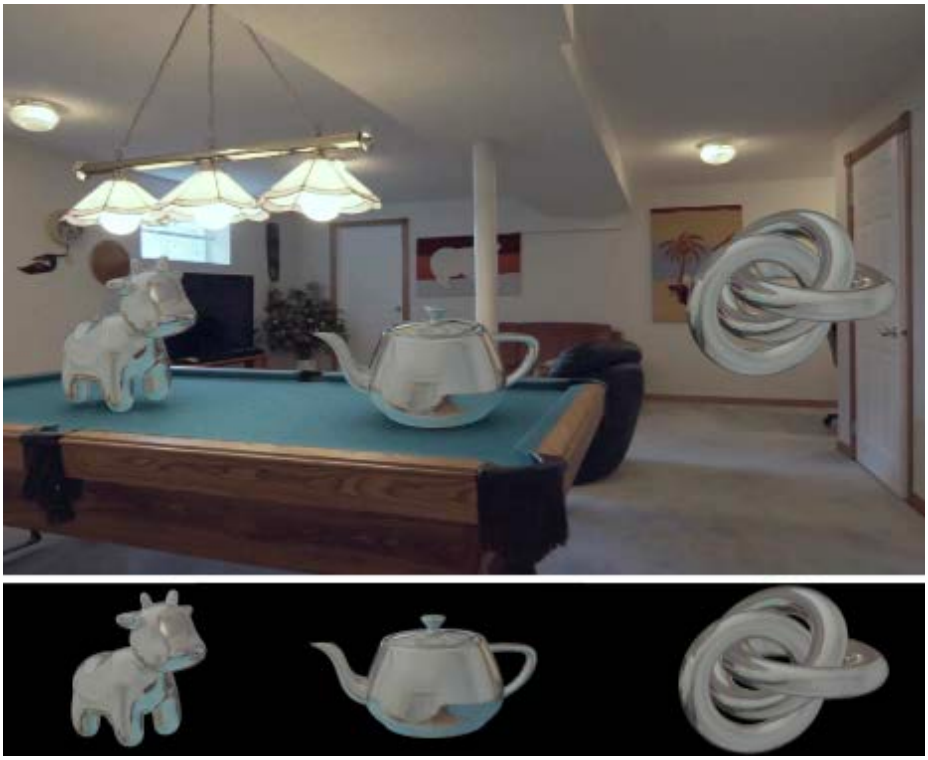


Figure 5.1. Our method predicts environment map lighting at any location in a 3D scene from a narrow-baseline stereo pair of images. We use this to convincingly insert specular objects into real photographs with spatially-coherent lighting that varies smoothly in 3D. Below, we isolate the relit objects to better visualize our estimated illumination. Notice how each inserted object contains different specular highlights and reflected colors corresponding to its 3D location, such as the light reflected on the cow's head and the corner of the table visible on the teapot.

5.1 Introduction

Rendering virtual objects into photographs of real scenes is a common task in mixed reality and image editing. Convincingly inserting such objects requires estimating both the geometry of the scene (so that inserted objects are correctly occluded by real scene content) as well as the incident illumination at points on each object's surface (so that inserted objects appear to be lit by the surrounding environment). The difficulty of this task is exacerbated by the fact that incident illumination can vary significantly across different locations within a scene, especially indoors, due to lights close to the inserted objects, shadows cast by scene geometry, and global illumination effects from nearby scene content. Additionally, compositing large objects, multiple objects, or objects that move within the scene is even more difficult as doing so requires estimating a spatially-varying model of illumination that is *spatially-coherent* (we (ab)use this term to mean that it varies smoothly as a function of position in accordance with a plausible 3D scene).

Current state-of-the-art algorithms for estimating global illumination either predict a single illumination for the entire scene [8, 34, 51, 71, 73, 116] or estimate spatially-varying illumination by separately predicting the lighting at individual 3D locations within the scene [36, 80, 127]. The single-illumination approach can only be used to illuminate small objects at a predefined location, while the separately-predicted spatially-varying approach can produce compelling results, but does not guarantee that the predicted illumination will vary smoothly as a function of position. In this work, we propose an algorithm that predicts a volumetric representation of the scene from a narrow-baseline stereo pair of images, and then uses that volumetric representation to produce a spatially-varying model of illumination by simply rendering that predicted volume from the set of required object insertion locations. Because our approach computes each environment map from a single predicted underlying volumetric scene representation using standard volume rendering, all estimated lighting is naturally consistent with the same 3D scene, and lighting locations can be queried at 100 frames per second allowing for real-time object insertion.

We specifically use a narrow-baseline stereo pair as input because: 1) multi-camera systems are ubiquitous in modern smartphones, 2) stereo enables us to estimate the high fidelity geometry required for simulating spatially-varying lighting effects due to observed scene content, and 3) we can leverage recent progress in using narrow-baseline stereo images to predict 3D scene representations for view synthesis [132, 166], enabling us to render novel views of the scene with relit objects for virtual reality object insertion.

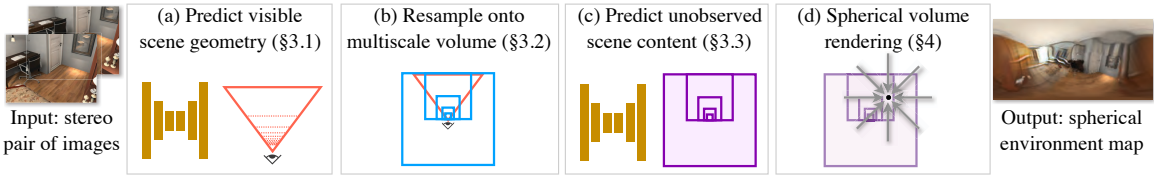


Figure 5.2. Our method takes a narrow-baseline stereo pair of images as input, uses a 3D CNN to predict an intermediate representation of visible scene geometry (a), resamples this onto a multiscale volume that encompasses unobserved regions of the scene (b), completes this volume with another 3D CNN (c), and renders spatially-coherent environment maps at any 3D location from this same volume using standard volume tracing (d).

To summarize, our primary technical contributions are:

1. A multiscale volumetric scene lighting representation that is specifically designed for estimating realistic spatially-varying lighting (Sec. 5.3.2) and a deep learning–based approach for predicting this representation using only a narrow-baseline stereo pair as input (Sec. 5.3.3). We design this representation to support rendering spatially-varying illumination without any network inference (Sec. 5.4), so lighting prediction is very fast and guaranteed to be spatially-coherent.

2. A training procedure that only needs perspective and panoramic views of scenes for supervision, instead of any ground-truth 3D scene representation (Sec. 5.5).

We demonstrate that estimating spatially-varying global illumination as a persistent 3D function quantitatively and qualitatively outperforms prior approaches. Our spatially-coherent estimated lighting can simulate convincing global illumination effects for rendering specular virtual objects moving within scenes.

5.2 Related Work

5.2.1 Estimating lighting from images

Inferring the intrinsic properties of lighting, materials, and geometry that together form an image is a fundamental problem that has been studied in various forms throughout the history of computer vision [5, 52]. Below, we review relevant prior works that use images to estimate representations of lighting for relighting virtual objects.

Seminal work by Debevec [20] showed that virtual objects can be convincingly inserted into real photographs by rendering virtual object models with high dynamic range (HDR) environment maps captured with bracketed exposures of a chrome ball. Many subsequent methods [8, 34, 51, 71, 73, 116] have demonstrated that machine learning techniques can be used to estimate an HDR environment map from a single low dynamic range (LDR) photograph.

However, a single environment map is insufficient for compositing multiple, large, or moving virtual objects into a captured scene, especially in indoor settings where light sources and other scene content may be close to the object insertion locations. To address this shortcoming, many works predict spatially-varying lighting from images by estimating a separate environment map for each pixel in the input image. Such approaches include algorithms designed specifically for spatially-varying lighting estimation [36] as well as methods that address the more general inverse rendering problem of jointly estimating the spatially-varying lighting, scene materials, and geometry that together produce an observed image [80, 118]. However, these approaches do not ensure that the illuminations predicted at different spatial locations correspond to a single 3D scene, and their approach of indexing lighting by image pixel coordinates cannot estimate lighting at locations other than points lying directly on visible scene surfaces. Karsch *et al*. [63] also address a similar inverse rendering problem, but instead estimate area lights in 3D by detecting visible light source locations and retrieving unobserved light sources from an annotated panorama database.

Our work is closely related to prior deep learning methods that estimate a portion of lighting in 3D. Neural Illumination [127] predicts the incident illumination at a location by first estimating per-pixel 3D geometry for the input image, reprojecting input image pixels into an environment map at the queried location, and finally using a 2D CNN to predict unobserved content in the resulting environment map. This strategy ensures spatial consistency for light emitted from scene

points that are visible in the input image (for which a single persistent geometry estimate is used), but because the environment map is separately completed for each lighting location using a 2D CNN, the lighting from unobserved scene points is not spatially-coherent. Recent work by Gardner *et al*. [33] trains a deep network to estimate the positions, intensities, and colors of a fixed number of light sources in 3D, along with an ambient light color. This ensures spatially-coherent lighting, but is unable to simulate realistic global illumination effects or light source occlusions, which can be very significant in indoor scenes, and therefore has difficulty rendering realistic specular objects. Furthermore, both of these methods require ground truth scene depths for training while our method only requires perspective and spherical panorama images.

5.2.2 Predicting 3D scene representations

Our strategy of estimating consistent spatially-varying lighting by predicting and rendering from a 3D scene representation is inspired by recent successes in using 3D representations for the image-based rendering problem of predicting novel views of a scene. Shum and Kang [120] provide an excellent review of classic approaches, ranging from light field rendering methods [76] that do not use any scene geometry, to texture mapping methods [21, 117] that use a global scene mesh. A key lesson from early work on image-based rendering is that more knowledge of scene geometry reduces the number of sampled images required for rendering new views [7, 9]. Modern approaches to view synthesis follow this lesson, rendering novel views by predicting representations of 3D scene geometry from sparsely-sampled collections of images. In particular, many recent methods predict layered or volumetric 3D scene representations, which have a regular grid structure that is well-suited to CNN pipelines. This includes algorithms for synthesizing novel outwards-facing views of large scenes [31, 89, 132, 166] and inwards-facing views of objects [84, 125, 155].

We adopt the approach of Zhou *et al*. [166] for learning to predict a layered representation of observed scene content. Their algorithm trains a CNN to predict a set of fronto-parallel RGBα planes sampled evenly in disparity within the camera frustum. Training proceeds by minimizing the difference between renderings of their model and held-out novel views, thereby obviating the need for ground truth 3D supervision. This representation, which they call a multiplane image (MPI), is closely related to representations used in volume rendering [24, 70, 74] and stereo matching [136]. Though this approach works well for view synthesis, it cannot be directly used for estimating spatially-varying lighting, as the majority of the illu-

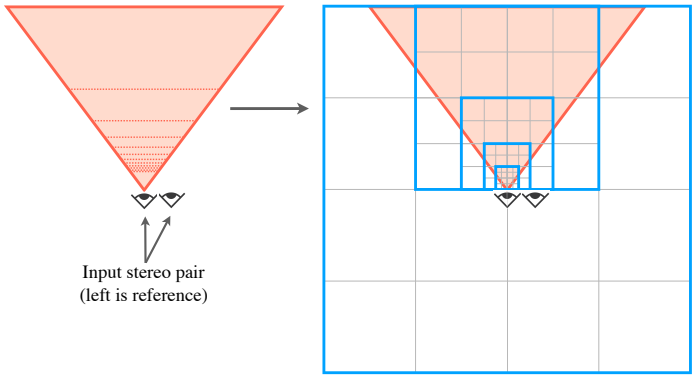
mination needed for relighting and compositing virtual objects often resides *outside* of the input image's field of view. We address this shortcoming by extending these models to predict a multiscale volumetric representation that includes scene content outside the input image's camera frustum, thereby allowing us to estimate incident illumination at any 3D location in the scene.

5.3 Multiscale Lighting Volume Prediction

Our goal is to take in a narrow-baseline pair of RGB images and associated camera poses, and output the incident illumination (represented as a spherical environment map) at any queried 3D location within the scene. Due to dataset limitations (see Sec. 5.5.2), we do not address LDR-to-HDR conversion and assume that the inputs are either HDR captures or that they can be converted to HDR by inverting a known tone mapping curve or by applying existing LDR-to-HDR conversion techniques [26].

We train a deep learning pipeline, visualized in Fig. 5.2, that regresses from the input image pair to a volumetric RGBα representation of the entire scene that includes areas outside of the reference camera frustum (we choose one of the two input images as the "reference" to be the center of our coordinate system), thereby allowing the illumination at any 3D location to be estimated by simply rendering the scene volume at that location. This representation enables us to reproduce effects such as: shadowing due to the occlusion of light sources by other scene content, realistic reflections on glossy and specular virtual objects, and color bleeding from the scene onto relit objects.

Naïvely representing an entire indoor scene as a dense high-resolution voxel grid is intractable due to memory constraints. Instead, we propose a multiscale volume representation designed to adequately sample the varying depth resolution provided by stereo matching, allocate sufficient resolution to areas where virtual objects would be inserted, and allocate lower resolution outside the observed field-of-view where scene content must be hallucinated. As shown in Fig. 5.2 our procedure for predicting this volume is: 1) A deep network predicts a layered representation of observed scene content from the input narrow-baseline stereo pair (Fig. 5.2a). 2) This layered representation is resampled onto a multiscale lighting volume that preserves the observed content representation's resolution (Fig. 5.2b). 3) Another deep network completes the multiscale lighting volume by hallucinating scene geometry and appearance outside the observed field of view (Fig. 5.2c).



(a) MPI sampling of visible scene content

(b) Multiscale volume sampling of entire scene

Figure 5.3. A 2D visualization of our 3D multiscale volume resampling. (a) First, given an input stereo pair of images, we predict scene content within the reference camera frustum as a set of RGBα planes spaced linearly in inverse depth. (b) Next, we resample the frustum geometry onto a set of nested cubes with increasingly finer sampling, centered around the input camera.

Once we generate a lighting volume for a scene, we can use classic volume rendering to estimate the incident illumination at any 3D location without requiring any network inference (Fig. 5.2d).

5.3.1 Observed content intermediate representation

We construct an intermediate representation of observed scene content as an MPI M, which consists of a set of fronto-parallel RGBα planes within the frustum of a reference camera, as visualized in Fig. 5.3a. As in Zhou *et al*. [166], we select one of the input images as a reference view and construct a plane sweep volume (PSV) in this frame for both input images. We concatenate these two PSVs along the channel dimension, forming an input tensor for a 3D encoder-decoder CNN that outputs an MPI, as suggested by follow-up works on MPI prediction [89, 132].

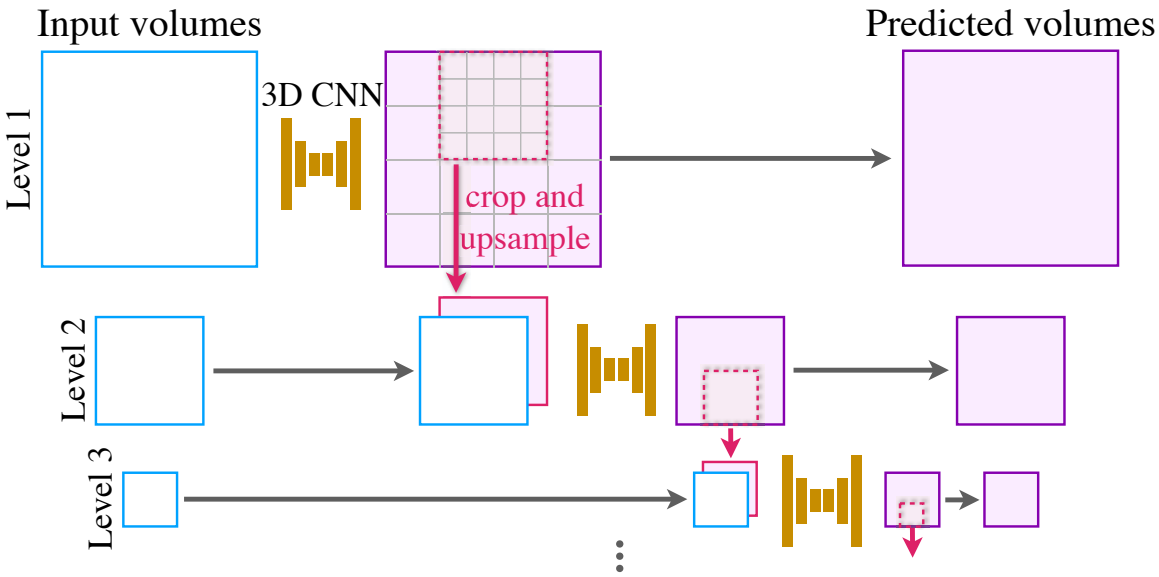


Figure 5.4. A visualization of our multiscale volume completion network. After resampling the visible scene geometry onto a series of nested cubes, we apply a volume completion network to hallucinate the unseen geometry at each level. The prediction is done in a coarse-to-fine manner, where the coarse prediction at level ℓ is cropped and upsampled then fed into the CNN along with the resampled visible volume to predict level $\ell + 1$.

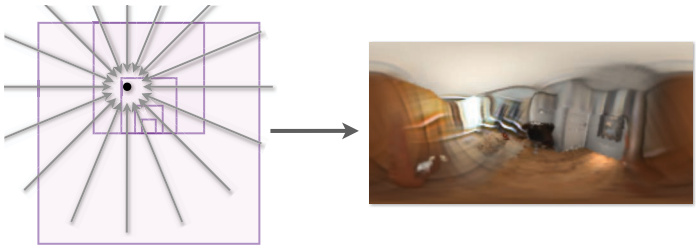
5.3.2 Multiscale volume resampling

This MPI provides an estimation of geometry for regions of the scene observed in the two input images. However, inserting a relit virtual object into the scene also requires estimating geometry and appearance for unobserved areas behind and to the sides of the input cameras' frustums, as visualized in Fig. 5.3, so our lighting representation must encompass this area. Furthermore, our volumetric lighting representation should allocate higher resolution to regions where we would insert objects in order to correctly render the larger movement of nearby content within environment maps as the queried location changes.

We design a multiscale volume lighting representation that encompasses both observed and unobserved regions, allocates finer resolution within the input field-of-view, and increases in resolution towards the front of the MPI frustum (which contains increasingly higher resolution estimated geometry from stereo matching). We initialize this multiscale volume by resampling the RGBα values from the MPI frustum onto a series of nested cubes $V^o = \{V_1^o, \ldots, V_L^o\}$ encompassing the whole scene, using trilinear interpolation. From the coarsest to finest level, each cube V_ℓ^o is half the spatial width of the previous level $V_{\ell-1}^o$ while maintaining the same grid resolution of 64^3. The largest, outermost cube V_1^o is centered at the first input camera pose and is wide enough to contain the whole MPI volume. Each smaller nested cube is offset such that the input camera pose lies at the back face of the cube. See Fig. 5.3 for a visualization of the MPI sampling pattern and how we resample it onto the multiscale volume structure. We find that this multiscale sampling pattern works well in practice for rendering convincing near-field lighting effects caused by scene geometry at our chosen environment map resolution of 120×240.

5.3.3 Multiscale volume completion

Now that we have a volumetric representation $V^o = \{V_1^o, \ldots, V_L^o\}$ of the entire scene that has been populated with observed content, we use a deep network to hallucinate the geometry and appearance of the unobserved content. We denote this "completed" multiscale volume by $V^c = \{V_1^c, \ldots, V_L^c\}$. We design a 3D CNN architecture that sequentially processes this multiscale volume from the coarsest to the finest resolution, predicting a completed volume V_ℓ^c at each resolution level. For each level, we first nearest-neighbor upsample the region of the previous coarser completed volume $V_{\ell-1}^c$ that overlaps the current level V_ℓ^o to 64^3 resolution. Then, we concatenate this to the current level's resampled volume V_ℓ^o along the channel dimension and use a 3D encoder-decoder CNN to predict the current level's com-



(a) Spherical volume tracing (b) Spherical environment map

Figure 5.5. Given our predicted multiscale lighting volume and a 3D location in the scene (shown as a black circle), we render the environment map by (a) tracing spherical rays through the volumes and alpha compositing from the outermost to the innermost RGBα value, producing a single spherical environment map (b).

pleted volume V_ℓ^c, with separate weights for each level. Figure 5.4 visualizes our coarse-to-fine network architecture.

5.4 Illumination Rendering

Given our multiscale lighting volume V^c, we estimate the illumination incident at any 3D location by using standard RGBα volume rendering [74] to generate a spherical environment map at that point (visualized in Fig. 5.5). In order to get the value at a pixel p for an environment map located at location $\mathbf{x}$, we must:

1. Generate the ray r (in world coordinates) that originates at $\mathbf{x}$ and intersects pixel p on the sphere, and
2. Trace r through the volume V^c, using alpha compositing to matte in the RGBα values as it intersects voxels from farthest to nearest.

As we trace r through V^c, we query the finest level defined at that location in space to ensure that predicted RGBα values at coarser levels never override predictions at finer levels. This rendering procedure is very fast since it does not involve network inference and is trivially parallelizable on GPUs, allowing us to render environment maps from a predicted multiscale volume at 100 frames per second.

5.5 Training and Dataset

Our model is trained end-to-end: a stereo pair is provided as input, the model renders a held-out novel view (sampled close to the reference view) from the intermediate MPI and a held-out environment map (sampled within the scene in front of the reference camera) from the completed multiscale lighting volume, and we update the model parameters only using the gradient of losses based on these two supervision images. This is possible since all steps in our pipeline are differentiable, including the multiscale volume resampling and the environment map rendering. Therefore, we do not require ground-truth geometry or other labels, in contrast to prior works in spatially-varying lighting estimation which either require scene geometry as supervision [36, 80, 116, 127] or for creating training data [33].

5.5.1 Training loss

The loss we minimize during training is the sum of an image reconstruction loss for rendering a held-out perspective view from our predicted MPI, an image reconstruction loss for rendering an environment map from our completed multiscale lighting volume, and an adversarial loss on the rendered environment map to encourage plausible high frequency content. For our reconstruction loss, we use a perceptual loss $\mathcal{L}_{vgg}$ based on features from a pre-trained VGG-19 network [123], as done by Chen and Koltun [12]. For our adversarial loss, we follow recent work in conditional image generation [102, 145] and use a PatchGAN [57] discriminator $\mathcal{D}$ with spectral normalization [91] and a hinge adversarial loss [81]. We train all networks in our pipeline by alternating between minimizing the reconstruction and adversarial losses with respect to the MPI prediction and volume completion networks' parameters:

$$\mathcal{L}_{train} = \mathcal{L}_{vgg}(i_r, i_{gt}) + \mathcal{L}_{vgg}(e_r, e_{gt}) - \mathcal{D}(e_r), \tag{5.1}$$

and minimizing the discriminator loss with respect to the discriminator network's parameters:

$$\mathcal{L}_{dis} = \max\left(0, 1 - \mathcal{D}\left(e_{gt}\right)\right) + \max\left(0, 1 + \mathcal{D}\left(e_r\right)\right), \tag{5.2}$$

where i_r, i_{gt}, e_r, e_{gt} are the rendered and ground truth perspective image and environment maps, respectively.

5.5.2 Dataset details

We train our model with photorealistic renderings of indoor scenes from the InteriorNet dataset [78]. We use 1634 of the provided camera sequences, each containing 1000 perspective projection images and 1000 spherical panorama images rendered along the same camera path. We reserve 10% of these sequences for our test set, and sample training examples from the remaining 90% of the sequences.

The images included in InteriorNet are not HDR, and unfortunately no equivalent dataset with HDR radiance values currently exists. In our experiments, we assume that the InteriorNet images can be treated as linear radiance values by applying an inverse gamma curve x^γ, with $\gamma = 2.2$.

To generate training examples from a sequence, we randomly sample three perspective projection images, evenly separated with a gap between 1 and 8 frames along the sequence's camera path, and a single spherical panorama image within 40 frames of the central perspective projection frame. Two of the perspective projection images are randomly selected to be the input to our model, while the third perspective image and the spherical panorama image are used as supervision. We reject training examples where the camera is closer than 0.1m from the scene, examples where adjacent perspective cameras are separated by less than 0.05m, and examples where the average pixel brightness is lower than 0.1. Additionally, we reject examples where the spherical panorama camera does not move more than the median scene depth into the scene, relative to the central perspective projection camera, so that the environment map locations we use for supervision are representative of realistic object insertion locations.

5.5.3 Additional details

We implement our full training pipeline in TensorFlow [1] and train our model on a single NVIDIA Tesla V100 GPU using the Adam optimizer [67] with a batch size of 1. For more stable training, we first pre-train the MPI prediction network for 240k iterations, then train both the MPI prediction network and volume completion networks with just image reconstruction losses for 450k iterations, and finally add the adversarial losses and train both networks along with a discriminator for an additional 30k iterations. We use an Adam step size of 10^{-4} for the first two stages and 10^{-5} for the third stage.

	Method	PSNR (dB) ↑	Angular Error (°) ↓
all content	DeepLight [73]	13.36 ± 1.29	7.15 ± 3.24
	Garon *et al.* [36]	13.21 ± 1.80	12.73 ± 7.17
	Neural Illumination [127]	16.59 ± 1.91	5.26 ± 2.84
	Ours (MPI only)	15.26 ± 2.11	6.41 ± 3.42
	Ours (no $\mathcal{L}_{adv}$)	17.54 ± 1.97	4.74 ± 2.70
	Ours	17.29 ± 1.97	4.71 ± 2.68
observed content	DeepLight [73]	13.94 ± 1.96	7.21 ± 4.05
	Garon *et al.* [36]	14.52 ± 2.30	12.33 ± 9.03
	Neural Illumination [127]	18.58 ± 3.55	4.97 ± 3.42
	Ours (MPI only)	17.97 ± 3.86	4.45 ± 4.46
	Ours (no $\mathcal{L}_{adv}$)	19.74 ± 3.64	4.18 ± 3.29
	Ours	19.79 ± 3.99	3.76 ± 3.09

Table 5.1. Quantitative results for rendered environment maps. We separately report performance on all content (the complete environment map) and on only observed content (the portion of each environment map that was observed in the input image). We report the PSNR and RGB angular error (following [73]) for the predictions for each method versus ground truth spherical environment maps.

5.6 Results

We validate the benefits of our algorithm by comparing our estimated environment maps to those of current state-of-the-art algorithms and ablated versions of our model.

For quantitative comparisons (Table 5.1), we sample a test set of 4950 examples (using the same training example rejection criteria described above in Sec. 5.5.2) from our InteriorNet test set, which consists of 163 camera sequences that were held out during training. Each example consists of two input images and one ground truth environment map that represents the lighting at a random 3D location within the reference camera frustum. We select a subset of these examples to show comparisons of virtual object insertion results in Fig. 5.6 and show additional insertion results for our method on real photographs in Fig. 5.7.

5.6.1 Comparisons to baseline methods

We compare our method to trained models of DeepLight [73] and Garon *et al*. [36] (which both take a single image as input) provided by the authors, and to a generous re-implementation of Neural Illumination [127] that has access to ground-truth geometry, in order to provide a fair comparison against our method which requires stereo input. Note that all quantitative metrics are computed on LDR images, due to the limitations of the InteriorNet dataset.

DeepLight [73] takes in a single image and outputs one HDR lighting map for the entire scene. The training data used for supervision is a set of three light probes placed 60cm in front of the camera; thus, this location is where the predicted lighting should be most accurate. The lighting is output in the form of a 32×32 HDR image of a mirror ball light probe. In order to compare these results with our method, we resample the mirror ball onto a higher resolution 120×240 spherical environment map, rotated to match the orientation of the target environment map. Because DeepLight does not predict spatially varying lighting, it underperforms our method on our test set of environment maps at various locations within the scene (see Table 5.1). Qualitatively, the limitations of a single environment map are apparent in relighting results since inserted objects cannot be correctly relit as they are moved around the scene.

Garon *et al*. [36] predicts spatially-varying lighting from a single image. Given a particular pixel on an object surface, the method is supervised to match the lighting

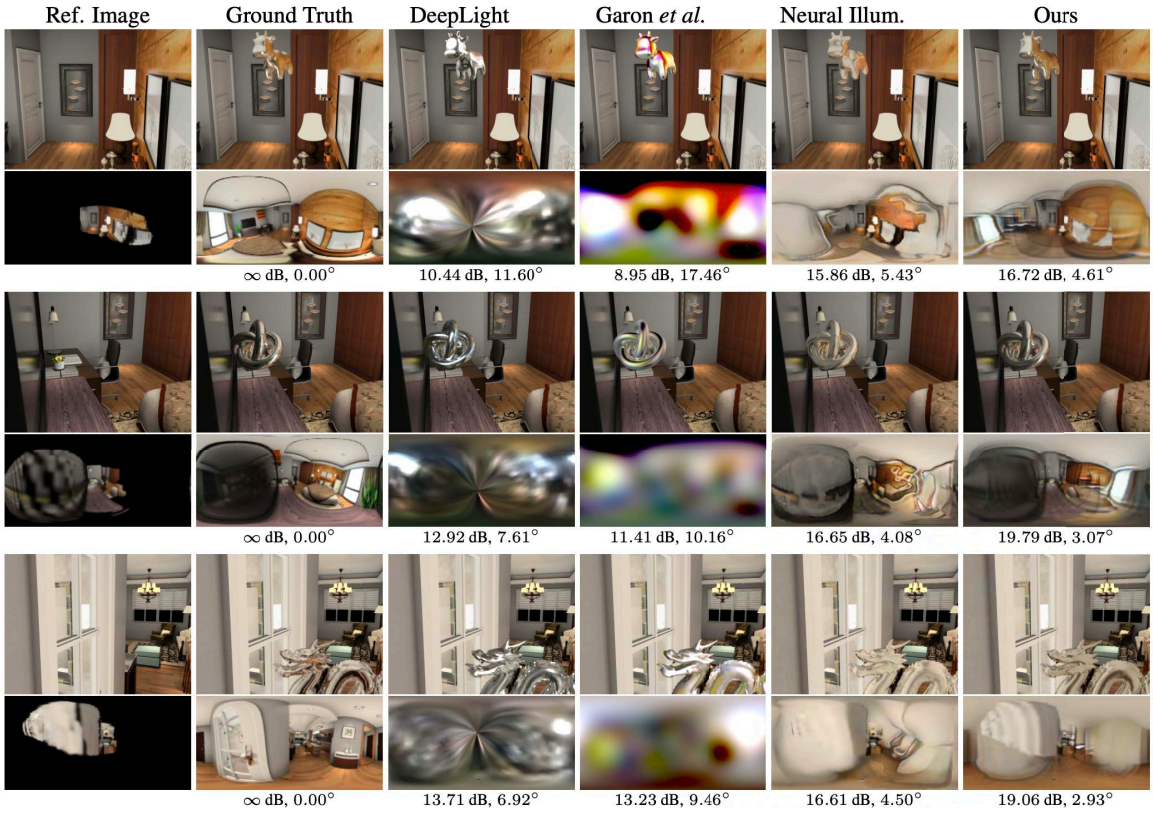


Figure 5.6. Estimated environment maps and images with inserted relit virtual objects for scenes from our synthetic InteriorNet test set. The leftmost column displays the input reference image (our method also takes a second input image) and the portion of the environment map that is visible in this reference image (used to visualize the portion of the environment map which must be hallucinated in black). We display quantitative metrics (PSNR and RGB Angular Error) for the predicted environment maps below each method's results. Our method outperforms all competing methods, both qualitatively and quantitatively, producing realistic environment maps with plausible unobserved regions. Inserted specular virtual objects relit with our environment maps have highlights and reflected colors that are closer to ground truth than those relit by baseline methods.

Figure 5.7. Real images from the RealEstate10K dataset [166] with inserted virtual objects relit with spatially-varying lighting estimated by our method. As in Fig. 5.1, we render perfectly specular objects by querying our multiscale lighting volume for spatially varying lighting values. The reflected colors are consistent with the scene content in the original image.

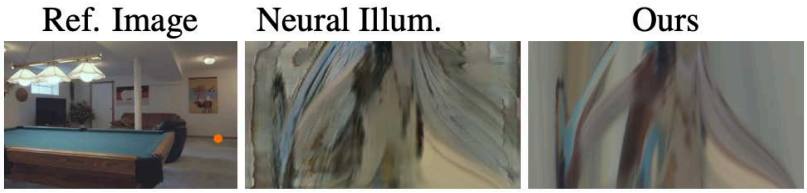


Figure 5.8. Our estimated illumination is more spatially-coherent than that of Neural Illumination [127]. Each row in the right two images is a row from environment maps rendered along the camera ray marked in orange (depth increases with lower rows). Our estimated illumination varies much more smoothly as a function of 3D position.

10cm away from that surface in the normal direction. This allows for some spatially-varying effects but heavily restricts the supported lighting locations. Additionally, this method predicts a low-dimensional representation of the environment map as 36 spherical harmonic coefficients for efficiency. Since our test set consists of higher resolution 120×240 environment maps sampled at locations that are not restricted to be near surfaces, this method performs significantly worse in our quantitative comparisons (see Table 5.1). Qualitatively, this lighting representation is sufficient for relighting diffuse objects, but its low resolution prevents it from plausibly relighting glossy and specular objects (see Fig. 5.6).

Neural Illumination [127] estimates lighting at any 3D scene point by first predicting a per-pixel geometry for a single input image, warping the input image with this geometry to render an incomplete environment map, using a 2D CNN to inpaint unobserved content, and finally using another 2D CNN to convert the environment map to HDR. We did not have access to the authors' original implementation, so we implemented a generous baseline that uses the *ground truth* depth to warp the visible scene content into an incomplete spherical environment map, then uses a 2D CNN to complete the lighting map. Despite having access to the ground truth geometry for the observed portion of the scene, this method is not guaranteed to produce spatially-coherent lighting predictions for the unobserved regions of a given scene, because the 2D completion CNN is run *independently* at each queried location. In contrast, our single multiscale 3D representation of the entire scene guarantees consistency across different 3D lighting locations. Figure 5.8 demonstrates how our method produces spatially-coherent lighting estimations that vary much more smoothly with 3D position than lighting estimated by Neural Illumination. Furthermore, Neural Illumination contains less realistic hallucinations of unobserved scene content, as shown in Fig. 5.6. This is because it is much harder for their 2D completion CNN to learn meaningful implicit priors on environment

maps since they can be observed with arbitrary rotations. In contrast, our strategy predicts lighting in a canonical 3D frame instead of in a 2D environment map pixel space, so it is much easier for our network to learn meaningful priors on the distribution of 3D lighting.

5.6.2 Comparisons to ablations of our method

We also present quantitative results from two ablations of our method in Table 5.1. Our "MPI only" ablation only uses our prediction of observed scene geometry to render the spherical environment map (and fills unobserved regions with grey). In this case, the network cannot add light values to the unseen parts of the scene, so the resulting environment maps are largely incomplete. Since these missing regions are the most important for relighting objects inserted into the scene, we see a significant decrease in quality. Interestingly, our full method even outperforms the "MPI only" ablation for the observed content, which shows that our multiscale volume completion network learns to correct errors in MPI prediction. Our "No $\mathcal{L}_{adv}$" ablation omits the adversarial loss when training the volume completion network. The resulting environment maps are slightly better quantitatively, but contain less high frequency detail, resulting in less realistic appearance when rendering glossy inserted objects.

5.7 Discussion

This chapter demonstrates that using a fixed volumetric 3D lighting model of the scene is a compelling strategy for estimating spatially-coherent illumination from images. We have chosen a multiscale volumetric lighting representation to make this approach tractable and proposed a deep learning pipeline to predict this lighting representation using only images as supervision. Our results demonstrate that this strategy produces plausible spatially-coherent lighting and outperforms prior state-of-the-art work.

However, we have just touched the surface of possible 3D lighting representations for this task. An exciting direction would be to develop models that adaptively allocate 3D samples as needed to represent a scene, rather than being limited to a fixed multiresolution sampling pattern. We hope that this work enables future

progress in predicting 3D scene representations for lighting estimation and other inverse rendering tasks.

In the next chapter, we present a volumetric 3D scene representation that is parameter-efficient and also adaptive in that the way it uses its parameters to represent scene content that is optimized in order to maximize the quality of rendered novel views.

Chapter 6

View Synthesis with Neural Radiance Fields

While the volumetric scene representations presented in the previous chapters have been effective for their respective view synthesis tasks, they are fundamentally constrained by their fixed sampling pattern. Consequentially, their ability to render novel views is restricted to the subset of views that their fixed sampling pattern was designed for. For example, the multiplane image representation is effective at synthesizing novel "forwards-facing" views, but is insufficient for rendering novel views that move closer to scene content or rotate with respect to the multiplane image frustum. Essentially, by fixing the sampling pattern before optimizing scene geometry, we have decided the maximum resolution of content in any portion of the scene. In this chapter, we present a radically different strategy for representing volumes in a paramter-efficient manner. Instead of representing scenes as discretized sampled volumes, we fit a continuous function approximator (in this case a deep neural network) to represent the scene's global volumetric function.

This chapter is based on joint work published at ECCV 2020 [90].

6.1 Introduction

In this work, we address the long-standing problem of view synthesis in a new way by directly optimizing parameters of a continuous 5D scene representation to minimize the error of rendering a set of captured images.

We represent a static scene as a continuous 5D function that outputs the radiance emitted in each direction (θ, ϕ) at each point (x, y, z) in space, and a density at each point which acts like a differential opacity controlling how much radiance is accumulated by a ray passing through (x, y, z). Our method optimizes a deep fully-connected neural network without any convolutional layers (often referred to as a multilayer perceptron or MLP) to represent this function by regressing from a single 5D coordinate (x, y, z, θ, ϕ) to a single volume density and view-dependent RGB color. To render this *neural radiance field* (NeRF) from a particular viewpoint we: 1) march camera rays through the scene to generate a sampled set of 3D points, 2) use those points and their corresponding 2D viewing directions as input to the neural network to produce an output set of colors and densities, and 3) use classical volume rendering techniques to accumulate those colors and densities into a 2D image. Because this process is naturally differentiable, we can use gradient descent to optimize this model by minimizing the error between each observed image and the corresponding views rendered from our representation. Minimizing this error across multiple views encourages the network to predict a coherent model of the scene by assigning high volume densities and accurate colors to the locations that contain the true underlying scene content. Figure 6.2 visualizes this overall pipeline.

We find that the basic implementation of optimizing a neural radiance field representation for a complex scene does not converge to a sufficiently high-resolution representation and is inefficient in the required number of samples per camera ray. We address these issues by transforming input 5D coordinates with a positional encoding that enables the MLP to represent higher frequency functions, and we propose a hierarchical sampling procedure to reduce the number of queries required to adequately sample this high-frequency scene representation.

Our approach inherits the benefits of volumetric representations: both can represent complex real-world geometry and appearance and are well suited for gradient-based optimization using projected images. Crucially, our method overcomes the prohibitive storage costs of *discretized* voxel grids when modeling complex scenes at high-resolutions. In summary, our technical contributions are:

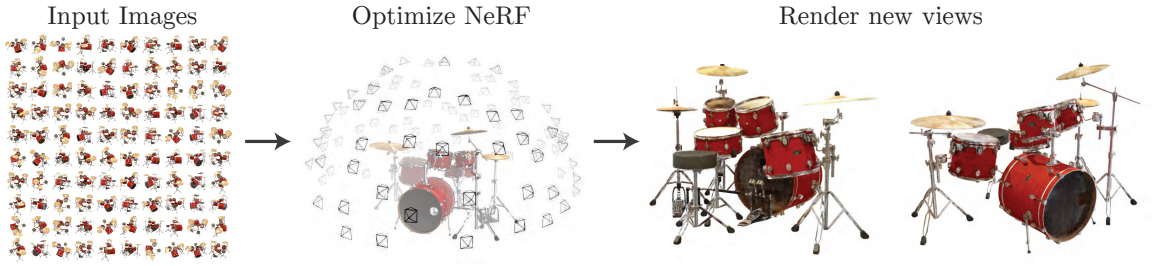


Figure 6.1. We present a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. We use techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, we visualize the set of 100 input views of the synthetic *Drums* scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from our optimized NeRF representation.

1. An approach for representing continuous scenes with complex geometry and materials as 5D neural radiance fields, parameterized as basic MLP networks.

2. A differentiable rendering procedure based on classical volume rendering techniques, which we use to optimize these representations from standard RGB images. This includes a hierarchical sampling strategy to allocate the MLP's capacity towards space with visible scene content.

3. A positional encoding to map each input 5D coordinate into a higher dimensional space, which enables us to successfully optimize neural radiance fields to represent high-frequency scene content.

We demonstrate that our resulting neural radiance field method quantitatively and qualitatively outperforms state-of-the-art view synthesis methods, including works that fit neural 3D representations to scenes as well as works that train deep convolutional networks to predict sampled volumetric representations. As far as we know, the work in this chapter presents the first continuous neural scene representation that is able to render high-resolution photorealistic novel views of real objects and scenes from RGB images captured in natural settings.

6.2 Related Work

A promising recent direction in computer vision is encoding objects and scenes in the weights of an MLP that directly maps from a 3D spatial location to an implicit representation of the shape, such as the signed distance [18] at that location. However, these methods have so far been unable to reproduce realistic scenes with complex geometry with the same fidelity as techniques that represent scenes using discrete representations such as triangle meshes or voxel grids. In this section, we review these two lines of work and contrast them with our approach, which enhances the capabilities of neural scene representations to produce state-of-the-art results for rendering complex realistic scenes.

A similar approach of using MLPs to map from low-dimensional coordinates to colors has also been used for representing other graphics functions such as images [134], textured materials [49, 96, 108, 109], and indirect illumination values [110].

Neural 3D shape representations Recent work has investigated the implicit representation of continuous 3D shapes as level sets by optimizing deep networks that map xyz coordinates to signed distance functions [59, 100] or occupancy fields [38, 88]. However, these models are limited by their requirement of access to ground truth 3D geometry, typically obtained from synthetic 3D shape datasets such as ShapeNet [128]. Subsequent work has relaxed this requirement of ground truth 3D shapes by formulating differentiable rendering functions that allow neural implicit shape representations to be optimized using only 2D images. Niemeyer *et al*. [94] represent surfaces as 3D occupancy fields and use a numerical method to find the surface intersection for each ray, then calculate an exact derivative using implicit differentiation. Each ray intersection location is provided as the input to a neural 3D texture field that predicts a diffuse color for that point. Sitzmann *et al*. [126] use a less direct neural 3D representation that simply outputs a feature vector and RGB color at each continuous 3D coordinate, and propose a differentiable rendering function consisting of a recurrent neural network that marches along each ray to decide where the surface is located.

Though these techniques can potentially represent complicated and high-resolution geometry, they have so far been limited to simple shapes with low geometric complexity, resulting in oversmoothed renderings. We show that an alternate strategy of optimizing networks to encode 5D radiance fields (3D volumes with 2D view-dependent appearance) can represent higher-resolution geometry and appearance to render photorealistic novel views of complex scenes.

View synthesis and image-based rendering Given a dense sampling of views, photorealistic novel views can be reconstructed by simple light field sample interpolation techniques [76, 17, 19]. For novel view synthesis with sparser view sampling, the computer vision and graphics communities have made significant progress by predicting traditional geometry and appearance representations from observed images. One popular class of approaches uses mesh-based representations of scenes with either diffuse [144] or view-dependent [7, 21, 151] appearance. Differentiable rasterizers [14, 37, 83, 85] or pathtracers [77, 95] can directly optimize mesh representations to reproduce a set of input images using gradient descent. However, gradient-based mesh optimization based on image reprojection is often difficult, likely because of local minima or poor conditioning of the loss landscape. Furthermore, this strategy requires a template mesh with fixed topology to be provided as an initialization before optimization [77], which is typically unavailable for unconstrained real-world scenes.

Another class of methods use volumetric representations to address the task of high-quality photorealistic view synthesis from a set of input RGB images. Volumetric approaches are able to realistically represent complex shapes and materials, are well-suited for gradient-based optimization, and tend to produce less visually distracting artifacts than mesh-based methods. Early volumetric approaches used observed images to directly color voxel grids [69, 115, 136]. More recently, several methods [30, 50, 62, 89, 103, 132, 142, 166] have used large datasets of multiple scenes to train deep networks that predict a sampled volumetric representation from a set of input images, and then use either alpha-compositing [105] or learned compositing along rays to render novel views at test time. Other works have optimized a combination of convolutional networks (CNNs) and sampled voxel grids for each specific scene, such that the CNN can compensate for discretization artifacts from low resolution voxel grids [125] or allow the predicted voxel grids to vary based on input time or animation controls [84]. While these volumetric techniques have achieved impressive results for novel view synthesis, their ability to scale to higher resolution imagery is fundamentally limited by poor time and space complexity due to their discrete sampling — rendering higher resolution images requires a finer sampling of 3D space. We circumvent this problem by instead encoding a *continuous* volume within the parameters of a deep fully-connected neural network, which not only produces significantly higher quality renderings than prior volumetric approaches, but also requires just a fraction of the storage cost of those *sampled* volumetric representations.

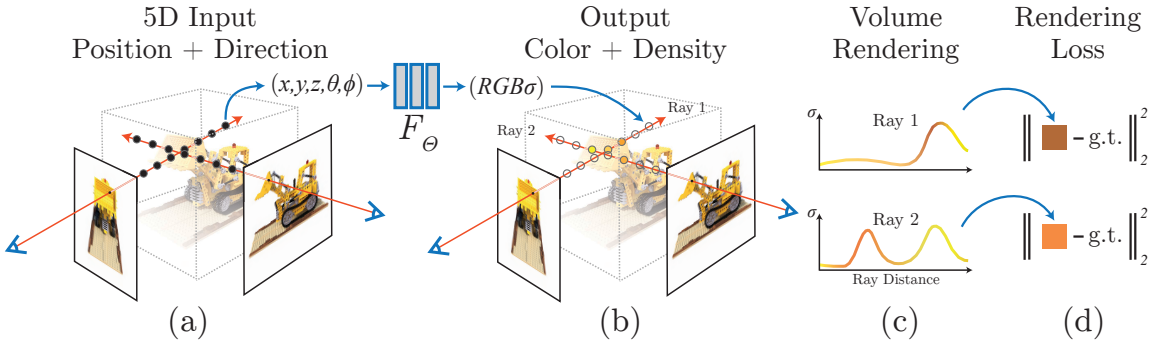


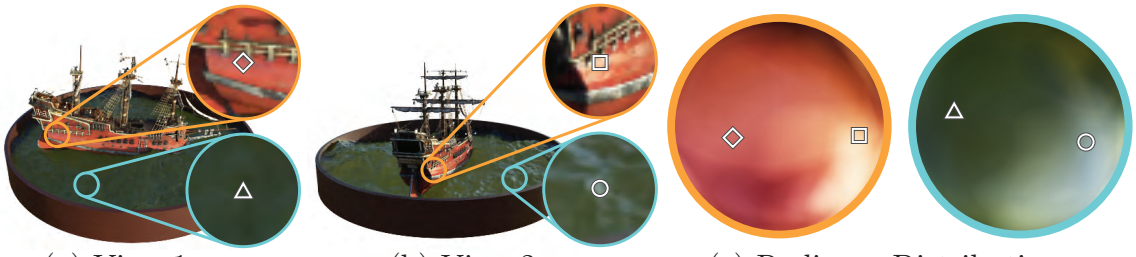
Figure 6.2. An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

6.3 Neural Radiance Field Scene Representation

We represent a continuous scene as a 5D vector-valued function whose input is a 3D location $\mathbf{x} = (x, y, z)$ and 2D viewing direction (θ, ϕ), and whose output is an emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. In practice, we express direction as a 3D Cartesian unit vector $\mathbf{d}$. We approximate this continuous 5D scene representation with an MLP network $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ and optimize its weights Θ to map from each input 5D coordinate to its corresponding volume density and directional emitted color.

We encourage the representation to be multiview consistent by restricting the network to predict the volume density σ as a function of only the location $\mathbf{x}$, while allowing the RGB color $\mathbf{c}$ to be predicted as a function of both location and viewing direction. To accomplish this, the MLP F_Θ first processes the input 3D coordinate $\mathbf{x}$ with 8 fully-connected layers (using ReLU activations and 256 channels per layer), and outputs σ and a 256-dimensional feature vector. This feature vector is then concatenated with the camera ray's viewing direction and passed to one additional fully-connected layer (using a ReLU activation and 128 channels) that output the view-dependent RGB color.

See Fig. 6.3 for an example of how our method uses the input viewing direction



(a) View 1 (b) View 2 (c) Radiance Distributions

Figure 6.3. A visualization of view-dependent emitted radiance. Our neural radiance field representation outputs RGB color as a 5D function of both spatial position $\mathbf{x}$ and viewing direction $\mathbf{d}$. Here, we visualize example directional color distributions for two spatial locations in our neural representation of the *Ship* scene. In (a) and (b), we show the appearance of two fixed 3D points from two different camera positions: one on the side of the ship (orange insets) and one on the surface of the water (blue insets). Our method predicts the changing specular appearance of these two 3D points, and in (c) we show how this behavior generalizes continuously across the whole hemisphere of viewing directions.

to represent non-Lambertian effects. As shown in Fig. 6.4, a model trained without view dependence (only $\mathbf{x}$ as input) has difficulty representing specularities.

6.4 Volume Rendering with Radiance Fields

Our 5D neural radiance field represents a scene as the volume density and directional emitted radiance at any point in space. We render the color of any ray passing through the scene using principles from classical volume rendering [60]. The volume density $\sigma(\mathbf{x})$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location $\mathbf{x}$. The expected color $C(\mathbf{r})$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \quad (6.1)$$

The function $T(t)$ denotes the accumulated transmittance along the ray from t_n to t, *i.e.*, the probability that the ray travels from t_n to t without hitting any other particle. Rendering a view from our continuous neural radiance field requires estimating

this integral $C(\mathbf{r})$ for a camera ray traced through each pixel of the desired virtual camera.

We numerically estimate this continuous integral using quadrature. Deterministic quadrature, which is typically used for rendering discretized voxel grids, would effectively limit our representation's resolution because the MLP would only be queried at a fixed discrete set of locations. Instead, we use a stratified sampling approach where we partition $[t_n, t_f]$ into N evenly-spaced bins and then draw one sample uniformly at random from within each bin:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right]. \tag{6.2}$$

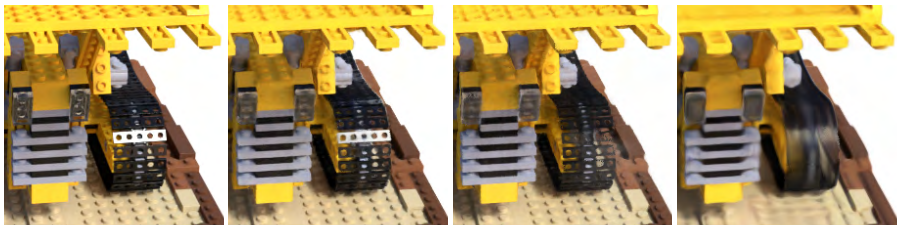
Although we use a discrete set of samples to estimate the integral, stratified sampling enables us to represent a continuous scene representation because it results in the MLP being evaluated at continuous positions over the course of optimization. We use these samples to estimate $C(\mathbf{r})$ with the quadrature rule discussed in the volume rendering review by Max [86]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,,\ \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right), \tag{6.3}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. This function for calculating $\hat{C}(\mathbf{r})$ from the set of $(\mathbf{c}_i, \sigma_i)$ values is trivially differentiable and reduces to traditional alpha compositing with alpha values $\alpha_i = 1 - \exp(-\sigma_i\delta_i)$.

6.5 Optimizing a Neural Radiance Field

In the previous section we have described the core components necessary for modeling a scene as a neural radiance field and rendering novel views from this representation. However, we observe that these components are not sufficient for achieving state-of-the-art quality, as demonstrated in Section 6.6.4). We introduce two improvements to enable representing high-resolution complex scenes. The first is a positional encoding of the input coordinates that assists the MLP in representing high-frequency functions, and the second is a hierarchical sampling procedure that allows us to efficiently sample this high-frequency representation.



Ground Truth Complete Model No View Dependence No Positional Encoding

Figure 6.4. Here we visualize how our full model benefits from representing view-dependent emitted radiance and from passing our input coordinates through a high-frequency positional encoding. Removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the positional encoding drastically decreases the model's ability to represent high frequency geometry and texture, resulting in an oversmoothed appearance.

6.5.1 Positional encoding

Despite the fact that neural networks are universal function approximators [53], we found that having the network F_Θ directly operate on $xyz\theta\phi$ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry. This is consistent with recent work by Rahaman *et al*. [107], which shows that deep networks are biased towards learning lower frequency functions. They additionally show that mapping the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation.

We leverage these findings in the context of neural scene representations, and show that reformulating F_Θ as a composition of two functions $F_\Theta = F'_\Theta \circ \gamma$, one learned and one not, significantly improves performance (see Fig. 6.4 and Table 6.2). Here γ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$, and F'_Θ is still simply a regular MLP. Formally, the encoding function we use is:

$$\gamma(p) = \big(\ \sin(2^0\pi p),\ \ \cos(2^0\pi p),\ \ \cdots,\ \ \sin\big(2^{L-1}\pi p\big),\ \ \cos\big(2^{L-1}\pi p\big)\ \big)\,. \tag{6.4}$$

This function $\gamma(\cdot)$ is applied separately to each of the three coordinate values in $\mathbf{x}$ (which are normalized to lie in $[-1, 1]$) and to the three components of the Cartesian viewing direction unit vector $\mathbf{d}$ (which by construction lie in $[-1, 1]$). In our experiments, we set $L = 10$ for $\gamma(\mathbf{x})$ and $L = 4$ for $\gamma(\mathbf{d})$.

A similar mapping is used in the popular Transformer architecture [143], where it is referred to as a *positional encoding*. However, Transformers use it for a different goal of providing the discrete positions of tokens in a sequence as input to an archi-

tecture that does not contain any notion of order. In contrast, we use these functions to map continuous input coordinates into a higher dimensional space to enable our MLP to more easily approximate a higher frequency function. Concurrent work on a related problem of modeling 3D protein structure from projections [165] also utilizes a similar input coordinate mapping.

6.5.2 Hierarchical volume sampling

Our rendering strategy of densely evaluating the neural radiance field network at N query points along each camera ray is inefficient: free space and occluded regions that do not contribute to the rendered image are still sampled repeatedly. We draw inspiration from early work in volume rendering [75] and propose a hierarchical representation that increases rendering efficiency by allocating samples proportionally to their expected effect on the final rendering.

Instead of just using a single network to represent the scene, we simultaneously optimize two networks: one "coarse" and one "fine". We first sample a set of N_c locations using stratified sampling, and evaluate the "coarse" network at these locations as described in Eqns. 6.2 and 6.3. Given the output of this "coarse" network, we then produce a more informed sampling of points along each ray where samples are biased towards the relevant parts of the volume. To do this, we first rewrite the alpha composited color from the coarse network $\hat{C}_c(\mathbf{r})$ in Eqn. 6.3 as a weighted sum of all sampled colors c_i along the ray:

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \,, \qquad w_i = T_i(1 - \exp(-\sigma_i \delta_i)) \,. \tag{6.5}$$

Normalizing these weights as $\hat{w}_i = {w_i}/{\sum_{j=1}^{N_c} w_j}$ produces a piecewise-constant PDF along the ray. We sample a second set of N_f locations from this distribution using inverse transform sampling, evaluate our "fine" network at the union of the first and second set of samples, and compute the final rendered color of the ray $\hat{C}_f(\mathbf{r})$ using Eqn. 6.3 but using all $N_c + N_f$ samples. This procedure allocates more samples to regions we expect to contain visible content. This addresses a similar goal as importance sampling, but we use the sampled values as a nonuniform discretization of the whole integration domain rather than treating each sample as an independent probabilistic estimate of the entire integral.

6.5.3 Implementation details

We optimize a separate neural continuous volume representation network for each scene. This requires only a dataset of captured RGB images of the scene, the corresponding camera poses and intrinsic parameters, and scene bounds (we use ground truth camera poses, intrinsics, and bounds for synthetic data, and use the COLMAP structure-from-motion package [113] to estimate these parameters for real data). At each optimization iteration, we randomly sample a batch of camera rays from the set of all pixels in the dataset, and then follow the hierarchical sampling described in Sec. 6.5.2 to query N_c samples from the coarse network and $N_c + N_f$ samples from the fine network. We then use the volume rendering procedure described in Sec. 6.4 to render the color of each ray from both sets of samples. Our loss is simply the total squared error between the rendered and true pixel colors for both the coarse and fine renderings:

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right] \tag{6.6}$$

where $\mathcal{R}$ is the set of rays in each batch, and $C(\mathbf{r})$, $\hat{C}_c(\mathbf{r})$, and $\hat{C}_f(\mathbf{r})$ are the ground truth, coarse volume predicted, and fine volume predicted RGB colors for ray $\mathbf{r}$ respectively. Note that even though the final rendering comes from $\hat{C}_f(\mathbf{r})$, we also minimize the loss of $\hat{C}_c(\mathbf{r})$ so that the weight distribution from the coarse network can be used to allocate samples in the fine network.

In our experiments, we use a batch size of 4096 rays, each sampled at $N_c = 64$ coordinates in the coarse volume and $N_f = 128$ additional coordinates in the fine volume. We use the Adam optimizer [67] with a learning rate that begins at 5×10^{-4} and decays exponentially to 5×10^{-5} over the course of optimization (other Adam hyperparameters are left at default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$). The optimization for a single scene typically take around 100–300k iterations to converge on a single NVIDIA V100 GPU (about 1–2 days).

6.6 Results

We quantitatively (Tables 6.1) and qualitatively (Figs. 6.5 and 6.6) show that our method outperforms prior work, and provide extensive ablation studies to validate our design choices (Table 6.2).

	Diffuse Synthetic 360° [125]			Realistic Synthetic 360°			Real Forward-Facing [89]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [126]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [84]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [89]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

Table 6.1. Our method quantitatively outperforms prior work on datasets of both synthetic and real images. We report PSNR/SSIM (higher is better) and LPIPS [163] (lower is better). The DeepVoxels [125] dataset consists of 4 diffuse objects with simple geometry. Our realistic synthetic dataset consists of pathtraced renderings of 8 geometrically complex objects with complex non-Lambertian materials. The real dataset consists of handheld forward-facing captures of 8 real-world scenes (NV cannot be evaluated on this data because it only reconstructs objects inside a bounded volume). Though LLFF achieves slightly better LPIPS, we urge readers to view our supplementary video where our method achieves better multiview consistency and produces fewer artifacts than all baselines.

6.6.1 Datasets

Synthetic renderings of objects We first show experimental results on two datasets of synthetic renderings of objects (Table 6.1, "Diffuse Synthetic 360°" and "Realistic Synthetic 360°"). The DeepVoxels [125] dataset contains four Lambertian objects with simple geometry. Each object is rendered at 512×512 pixels from viewpoints sampled on the upper hemisphere (479 as input and 1000 for testing). We additionally generate our own dataset containing pathtraced images of eight objects that exhibit complicated geometry and realistic non-Lambertian materials. Six are rendered from viewpoints sampled on the upper hemisphere, and two are rendered from viewpoints sampled on a full sphere. We render 100 views of each scene as input and 200 for testing, all at 800×800 pixels.

Real images of complex scenes We show results on complex real-world scenes captured with roughly forward-facing images (Table 6.1, "Real Forward-Facing"). This dataset consists of 8 scenes captured with a handheld cellphone (5 taken from the LLFF paper and 3 that we capture), captured with 20 to 62 images, and hold out $1/8$ of these for the test set. All images are 1008×756 pixels.

Figure 6.5. Comparisons on test-set views for scenes from our new synthetic dataset generated with a physically-based renderer. Our method is able to recover fine details in both geometry and appearance, such as *Ship*'s rigging, *Lego*'s gear and treads, *Microphone*'s shiny stand and mesh grille, and *Material*'s non-Lambertian reflectance. LLFF exhibits banding artifacts on the *Microphone* stand and *Material*'s object edges and ghosting artifacts in *Ship*'s mast and inside the *Lego* object. SRN produces blurry and distorted renderings in every case. Neural Volumes cannot capture the details on the *Microphone*'s grille or *Lego*'s gears, and it completely fails to recover the geometry of *Ship*'s rigging.

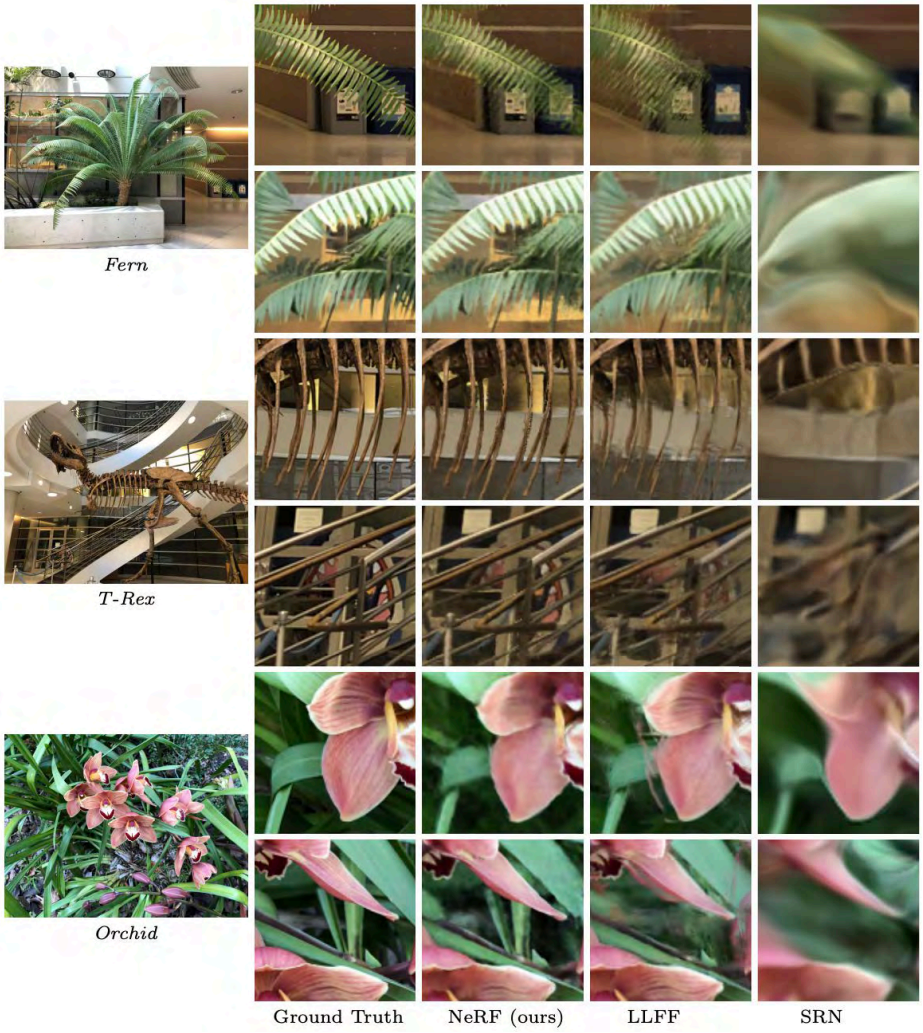


Figure 6.6. Comparisons on test-set views of real world scenes. LLFF is specifically designed for this use case (forward-facing captures of real scenes). Our method is able to represent fine geometry more consistently across rendered views than LLFF, as shown in *Fern*'s leaves and the skeleton ribs and railing in *T-rex*. Our method also correctly reconstructs partially occluded regions that LLFF struggles to render cleanly, such as the yellow shelves behind the leaves in the bottom *Fern* crop and green leaves in the background of the bottom *Orchid* crop. Blending between multiples renderings can also cause repeated edges in LLFF, as seen in the top *Orchid* crop. SRN captures the low-frequency geometry and color variation in each scene but is unable to reproduce any fine detail.

6.6.2 Comparisons

To evaluate our model we compare against current top-performing techniques for view synthesis, detailed below. All methods use the same set of input views to train a separate network for each scene except Local Light Field Fusion [89], which trains a single 3D convolutional network on a large dataset, then uses the same trained network to process input images of new scenes at test time.

Neural Volumes (NV) [84] synthesizes novel views of objects that lie entirely within a bounded volume in front of a distinct background (which must be separately captured without the object of interest). It optimizes a deep 3D convolutional network to predict a discretized RGBα voxel grid with 128^3 samples as well as a 3D warp grid with 32^3 samples. The algorithm renders novel views by marching camera rays through the warped voxel grid.

Scene Representation Networks (SRN) [126] represent a continuous scene as an opaque surface, implicitly defined by a MLP that maps each (x, y, z) coordinate to a feature vector. They train a recurrent neural network to march along a ray through the scene representation by using the feature vector at any 3D coordinate to predict the next step size along the ray. The feature vector from the final step is decoded into a single color for that point on the surface. Note that SRN is a better-performing followup to DeepVoxels [125] by the same authors, which is why we do not include comparisons to DeepVoxels.

Local Light Field Fusion (LLFF) [89] LLFF is designed for producing photorealistic novel views for well-sampled forward facing scenes. It uses a trained 3D convolutional network to directly predict a discretized frustum-sampled RGBα grid (multiplane image or MPI [166]) for each input view, then renders novel views by alpha compositing and blending nearby MPIs into the novel viewpoint.

6.6.3 Analysis

We thoroughly outperform both baselines that also optimize a separate network per scene (NV and SRN) in all scenarios. Furthermore, we produce qualitatively and quantitatively superior renderings compared to LLFF (across all except one metric) while using only their input images as our entire training set.

The SRN method produces heavily smoothed geometry and texture, and its representational power for view synthesis is limited by selecting only a single depth and color per camera ray. The NV baseline is able to capture reasonably detailed volumetric geometry and appearance, but its use of an underlying explicit 128^3 voxel grid prevents it from scaling to represent fine details at high resolutions. LLFF specifically provides a "sampling guideline" to not exceed 64 pixels of disparity between input views, so it frequently fails to estimate correct geometry in the synthetic datasets which contain up to 400-500 pixels of disparity between views. Additionally, LLFF blends between different scene representations for rendering different views, resulting in perceptually-distracting inconsistency.

The biggest practical tradeoffs between these methods are time versus space. All compared single scene methods take at least 12 hours to train per scene. In contrast, LLFF can process a small input dataset in under 10 minutes. However, LLFF produces a large 3D voxel grid for every input image, resulting in enormous storage requirements (over 15GB for one "Realistic Synthetic" scene). Our method requires only 5 MB for the network weights (a relative compression of $3000\times$ compared to LLFF), which is even less memory than the *input images alone* for a single scene from any of our datasets.

6.6.4 Ablation studies

We validate our algorithm's design choices and parameters with an extensive ablation study in Table 6.2. We present results on our "Realistic Synthetic 360°" scenes. Row 9 shows our complete model as a point of reference. Row 1 shows a minimalist version of our model without positional encoding (PE), view-dependence (VD), or hierarchical sampling (H). In rows 2–4 we remove these three components one at a time from the full model, observing that positional encoding (row 2) and view-dependence (row 3) provide the largest quantitative benefit followed by hierarchical sampling (row 4). Rows 5–6 show how our performance decreases as the number of input images is reduced. Note that our method's performance using only 25 input images still exceeds NV, SRN, and LLFF across all metrics when they are provided with 100 images. In rows 7–8 we validate our choice of the maximum frequency L used in our positional encoding for $\mathbf{x}$ (the maximum frequency used for $\mathbf{d}$ is scaled proportionally). Only using 5 frequencies reduces performance, but increasing the number of frequencies from 10 to 15 does not improve performance. We believe the benefit of increasing L is limited once 2^L exceeds the maximum frequency present in the sampled input images (roughly 1024 in our data).

	Input	#Im.	L	(N_c , N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

Table 6.2. An ablation study of our model. Metrics are averaged over the 8 scenes from our realistic synthetic dataset. See Sec. 6.6.4 for detailed descriptions.

6.7 Discussion

Our work directly addresses deficiencies of prior work that uses MLPs to represent objects and scenes as continuous functions. We demonstrate that representing scenes as 5D neural radiance fields (an MLP that outputs volume density and view-dependent emitted radiance as a function of 3D location and 2D viewing direction) produces better renderings than the previously-dominant approach of training deep convolutional networks to output discretized voxel representations.

Although we have proposed a hierarchical sampling strategy to make rendering more sample-efficient (for both training and testing), there is still much more progress to be made in investigating techniques to efficiently optimize and render neural radiance fields. Another direction for future work is interpretability: sampled representations such as voxel grids and meshes admit reasoning about the expected quality of rendered views and failure modes, but it is unclear how to analyze these issues when we encode scenes in the weights of a deep neural network. We believe that this work makes progress towards a graphics pipeline based on real world imagery, where complex scenes could be composed of neural radiance fields optimized from images of actual objects and scenes.

Chapter 7

Conclusion

In this dissertation, we have shown that using observed images to recover a persistent volumetric 3D model of the scene is a compelling strategy for creating a digital representation of real-world objects and scenes that can be rendered from novel unobserved viewpoints, just like hand-crafted 3D graphics models. Two key contributions that enabled our success were designing volumetric scene representations that are parameter-efficient, and optimizing these representations end-to-end to maximize the fidelity of the rendered novel views.

An important lesson I have learned in my work on view synthesis is the extent to which our tools shape our thinking. This has been a prevalent theme throughout the history of view synthesis. In the pre-deep-learning era, view synthesis algorithms for sparsely-captured images were largely designed to overcome the flaws in geometry estimated by off-the-shelf structure-from-motion, multiview stereo, and meshing algorithms. Similarly, early works applying deep learning to view synthesis were largely designed to use regularly-sampled grid representations in order to fit into the array programming paradigm popularized by deep learning software frameworks. I hope that the overall theme of this dissertation, rethinking the scene representations we use for the task of view synthesis, helps emphasize the importance of realizing the limitations of our current tools and frameworks and encourages the research community to pursue solutions based on their fundamental attributes.

While the work presented in this dissertation has substantially increased the capability of novel view synthesis algorithms, this functionality alone does not fully

capture everything we desire from 3D graphics assets. However, our work has made important progress to this goal, and we are excited to see how future work builds upon the representations we have developed to realize this grand goal. Indeed, the neural radiance field representation discussed in Chapter 6 has already inspired follow-up works that extend the representation to enable relighting [130] and dynamic scenes [79, 101]. It would be interesting to further extend this representation to enable physical simulation and rigging for user-controlled animation.

While this dissertation has shown that volumetric representations provide compelling benefits for view synthesis, these benefits primarily enable efficient and effective recovery of scene geometry from images, at the cost of efficient rendering after the scene representation has been recovered. Put another way, the volumetric scene representations presented in this paper are most effective for the computer vision goal of estimating a scene representation, as opposed to the computer graphics goal of rendering a given scene representation. Furthermore, the modern computer graphics pipeline has been largely developed around triangle meshes, and there are many tools and techniques built for editing mesh-based geometry that are not easily applied to volumetric geometry. I think that a potentially fruitful direction for future work would be to explore hybrid representations that combine the optimization benefits of volumes with the efficient rendering, editing, and simulation benefits of surfaces.

Finally, I look forward to the day when anyone can easily create convincing digital 3D content from photos of scenes and objects, and seamlessly use this real world content in digital art, video games, photographs, and films.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[2] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, "Jump: Virtual reality video," *SIGGRAPH Asia*, 2016.

[3] S.-H. Baek, I. Choi, and M. H. Kim, "Multiview image completion with space structure propagation," *CVPR*, 2016.

[4] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[5] H. G. Barrow and J. M. Tenenbaum, "Recovering intrinsic scene characteristics from images," *Computer Vision Systems*, 1978.

[6] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," *Proceedings of SIGGRAPH*, 2000.

[7] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," *SIGGRAPH*, 2001.

[8] D. A. Calian, J. Lalonde, P. F. U. Gotardo, T. Simon, I. A. Matthews, and K. Mitchell, "From Faces to Outdoor Light Probes," *Computer Graphics Forum*, 2018.

[9] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," *SIGGRAPH*, 2000.

[10] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv:1512.03012*, 2015.

[11] G. Chaurasia, S. Duchêne, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *SIGGRAPH*, 2013.

[12] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," *ICCV*, 2017.

[13] S. E. Chen and L. Williams, "View interpolation for image synthesis," *SIGGRAPH*, 1993.

[14] W. Chen, J. Gao, H. Ling, E. J. Smith, J. Lehtinen, A. Jacobson, and S. Fidler, "Learning to predict 3D objects with an interpolation-based differentiable renderer," *NeurIPS*, 2019.

[15] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," *ECCV*, 2016.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *ICLR*, 2016.

[17] M. Cohen, S. J. Gortler, R. Szeliski, R. Grzeszczuk, and R. Szeliski, "The lumigraph," *SIGGRAPH*, 1996.

[18] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *SIGGRAPH*, 1996.

[19] A. Davis, M. Levoy, and F. Durand, "Unstructured light fields," *Computer Graphics Forum*, 2012.

[20] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," *SIGGRAPH*, 2008.

[21] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," *SIGGRAPH*, 1996.

[22] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari, "Peeking behind objects: Layered depth prediction from a single image," *arXiv:1807.08776*, 2018.

[23] P. Didyk, P. Sitthi-Amorn, W. T. Freeman, F. Durand, and W. Matusik, "3dtv at home: Eulerian-lagrangian stereo-to-multiview conversion," *SIGGRAPH Asia*, 2013.

[24] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," *SIGGRAPH*, 1988.

[25] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *ICCV*, 2015.

[26] G. Eilertsen, J. Kronander, G. Denes, R. Mantiuk, and J. Unger, "HDR image reconstruction from a single exposure using deep CNNs," *SIGGRAPH Asia*, 2017.

[27] S. M. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, *et al.*, "Neural scene representation and rendering," *Science*, 2018.

[28] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," *CVPR*, 2017.

[29] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," *CVPR*, 2016.

[30] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker, "DeepView: view synthesis with learned gradient descent," *CVPR*, 2019.

[31] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," *CVPR*, 2016.

[32] W. T. Freeman, "Exploiting the generic viewpoint assumption," *IJCV*, 1996.

[33] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagne, and J.-F. Lalonde, "Deep parametric indoor lighting estimation," *ICCV*, 2019.

[34] M.-A. Gardner, K. Sunkavalli, E. Yumer, X. Shen, E. Gambaretto, C. Gagné, and J.-F. Lalonde, "Learning to predict indoor illumination from a single image," *SIGGRAPH Asia*, 2017.

[35] R. Garg, V. Kumar BG, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: geometry to the rescue," *ECCV*, 2016.

[36] M. Garon, K. Sunkavalli, S. Hadap, N. Carr, and J.-F. Lalonde, "Fast spatially-varying indoor lighting estimation," *CVPR*, 2019.

[37] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, , and W. T. Freeman, "Unsupervised training for 3D morphable model regression," *CVPR*, 2018.

[38] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," *CVPR*, 2020.

[39] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," *ECCV*, 2016.

[40] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *CVPR*, 2017.

[41] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski, "Ambient point clouds for view interpolation," *ACM Transactions on Graphics*, 2010.

[42] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," *SIGGRAPH*, 1996.

[43] C. Guillemot and O. L. Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Processing Magazine*, 2014.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning," *CVPR*, 2016.

[45] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, "Casual 3D Photography," *SIGGRAPH Asia*, 2017.

[46] P. Hedman and J. Kopf, "Instant 3D photography," *SIGGRAPH*, 2018.

[47] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *SIGGRAPH Asia*, 2018.

[48] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable Inside-Out Image-Based Rendering," *SIGGRAPH Asia*, 2016.

[49] P. Henzler, N. J. Mitra, and T. Ritschel, "Learning a neural 3D texture space from 2D exemplars," *CVPR*, 2020.

[50] P. Henzler, V. Rasche, T. Ropinski, and T. Ritschel, "Single-image tomography: 3d volumes from 2d cranial x-rays," *Eurographics*, 2018.

[51] Y. Hold-Geoffroy, K. Sunkavalli, S. Hadap, E. Gambaretto, and J.-F. Lalonde, "Deep outdoor illumination estimation," *CVPR*, 2017.

[52] B. K. P. Horn, "Determining lightness from an image," *Computer Graphics and Image Processing*, 1974.

[53] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, 1989.

[54] J. Howard, B. S. Morse, S. Cohen, and B. L. Price, "Depth-based patch scaling for content-aware stereo image completion," *WACV*, 2014.

[55] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "Deepmvs: Learning multi-view stereopsis," *CVPR*, 2018.

[56] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *Journal of Machine Learning Research*, 2015.

[57] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[58] H.-G. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y.-W. Tai, and I. S. Kweon, "Accurate depth map estimation from a lenslet light field camera," *CVPR*, 2015.

[59] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3d scenes," *CVPR*, 2020.

[60] J. T. Kajiya and B. P. V. Herzen, "Ray tracing volume densities," *Computer Graphics (SIGGRAPH)*, 1984.

[61] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *SIGGRAPH Asia*, 2016.

[62] A. Kar, C. Häne, and J. Malik, "Learning a multi-view stereo machine," *NeurIPS*, 2017.

[63] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth, "Automatic scene inference for 3D object compositing," *SIGGRAPH*, 2014.

[64] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *SIGGRAPH*, 2013.

[65] P. Kellnhofer, P. Didyk, S.-P. Wang, P. Sitthi-Amorn, W. Freeman, F. Durand, and W. Matusik, "3DTV at home: Eulerian-lagrangian stereo-to-multiview conversion," *SIGGRAPH*, 2017.

[66] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *ICCV*, 2017.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[68] J. Kopf, F. Langguth, D. Scharstein, R. Szeliski, and M. Goesele, "Image-based rendering in the gradient domain," *SIGGRAPH Asia*, 2013.

[69] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, 2000.

[70] P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," *SIGGRAPH*, 1994.

[71] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan, "Estimating natural illumination from a single outdoor image," *ICCV*, 2009.

[72] D. Lanman, R. Raskar, A. Agrawal, and G. Taubin, "Shield fields: Modeling and capturing 3D occluders," *SIGGRAPH Asia*, 2008.

[73] C. LeGendre, W.-C. Ma, G. Fyffe, J. Flynn, L. Charbonnel, J. Busch, and P. Debevec, "Deeplight: Learning illumination for unconstrained mobile mixed reality," *CVPR*, 2019.

[74] M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, 1988.

[75] ——, "Efficient ray tracing of volume data," *ACM Transactions on Graphics*, 1990.

[76] M. Levoy and P. Hanrahan, "Light field rendering," *SIGGRAPH*, 1996.

[77] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," *ACM Transactions on Graphics* (*SIGGRAPH Asia*), 2018.

[78] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "InteriorNet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," *BMVC*, 2018.

[79] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," *arXiv*, 2020.

[80] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image," *arXiv:1905.02722*, 2019.

[81] J. H. Lim and J. C. Ye, "Geometric GAN," *arXiv:1705.02894*, 2017.

[82] G. Lippmann, "La photographie intégrale," *Comptes-Rendus, Académie des Sciences*, 1908.

[83] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3D reasoning," *ICCV*, 2019.

[84] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *ACM Transactions on Graphics (SIGGRAPH)*, 2019.

[85] M. M. Loper and M. J. Black, "OpenDR: An approximate differentiable renderer," *ECCV*, 2014.

[86] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, 1995.

[87] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," *SIGGRAPH*, 1995.

[88] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," *CVPR*, 2019.

[89] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (SIGGRAPH)*, 2019.

[90] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *ECCV*, 2020.

[91] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *ICLR*, 2018.

[92] R. Ng and P. Hanrahan, "Digital correction of lens aberrations in light field photography." *SPIE International Optical Design*, 2006.

[93] R. Ng, "Fourier slice photography," *ACM Transactions on Graphics* (*SIGGRAPH*), 2005.

[94] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision," *CVPR*, 2019.

[95] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *ACM Transactions on Graphics* (*SIGGRAPH Asia*), 2019.

[96] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger, "Texture fields: Learning texture representations in function space," *ICCV*, 2019.

[97] R. Ortiz-Cayon, A. Djelouah, and G. Drettakis, "A bayesian approach for selective image-based rendering using superpixels," *International Conference on 3D Vision* (*3DV*), 2015.

[98] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec, "A system for acquiring, processing, and rendering panoramic light field stills for virtual reality," *SIGGRAPH Asia*, 2018.

[99] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, "Transformation-grounded image generation network for novel 3D view synthesis," *ECCV*, 2016.

[100] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," *CVPR*, 2019.

[101] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Deformable neural radiance fields," *arXiv*, 2020.

[102] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," *CVPR*, 2019.

[103] E. Penner and L. Zhang, "Soft 3D reconstruction for view synthesis," *ACM Transactions on Graphics* (*SIGGRAPH Asia*), 2017.

[104] J. Philip and G. Drettakis, "Plane-based multi-view inpainting for image-based rendering in large scenes," *I3D*, 2018.

[105] T. Porter and T. Duff, "Compositing digital images," *Computer Graphics (SIGGRAPH)*, 1984.

[106] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, Y. Wang, and A. Yuille, "Unrealcv: Virtual worlds for computer vision," *ACM Multimedia Open Source Software Competition*, 2017.

[107] N. Rahaman, A. Baratin, D. Arpit, F. Dräxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. C. Courville, "On the spectral bias of neural networks," *ICML*, 2018.

[108] G. Rainer, A. Ghosh, W. Jakob, and T. Weyrich, "Unified neural encoding of BTFs," *Computer Graphics Forum (Eurographics)*, 2020.

[109] G. Rainer, W. Jakob, A. Ghosh, and T. Weyrich, "Neural BTF compression and interpolation," *Computer Graphics Forum (Eurographics)*, 2019.

[110] P. Ren, J. Wang, M. Gong, S. Lin, X. Tong, and B. Guo, "Global illumination with radiance regression functions," *ACM Transactions on Graphics*, 2013.

[111] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3D structure from images," *NIPS*, 2016.

[112] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: learning 3-D scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[113] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," *CVPR*, 2016.

[114] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," *ECCV*, 2016.

[115] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, 1999.

[116] S. Sengupta, J. Gu, K. Kim, G. Liu, D. W. Jacobs, and J. Kautz, "Neural inverse rendering of an indoor scene from a single image," *ICCV*, 2019.

[117] J. Shade, S. J. Gortler, L.-W. He, and R. Szeliski, "Layered depth images," *SIGGRAPH*, 1998.

[118] E. Shelhamer, J. T. Barron, and T. Darrell, "Scene intrinsics and depth from a single image," *ICCV Workshops*, 2015.

[119] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand, "Light field reconstruction using sparsity in the continuous Fourier domain," *ACM Transactions on Graphics*, 2015.

[120] H.-Y. Shum, S.-C. Chan, and S. B. Kang, *Image-Based Rendering*. Springer, 2006.

[121] H.-Y. Shum and S. B. Kang, "A review of image-based rendering techniques," *Proceedings of Visual Communications and Image Processing*, 2000.

[122] E. Simoncelli, "Statistical models for images:compression restoration and synthesis," *Asilomar Conference on Signals, Systems, and Computers*, 1997.

[123] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[124] S. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski, "Image-based rendering for scenes with reflections," *SIGGRAPH*, 2012.

[125] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer, "Deepvoxels: Learning persistent 3D feature embeddings," *CVPR*, 2019.

[126] V. Sitzmann, M. Zollhoefer, and G. Wetzstein, "Scene representation networks: Continuous 3D-structure-aware neural scene representations," *NeurIPS*, 2019.

[127] S. Song and T. Funkhouser, "Neural illumination: Lighting prediction for indoor environments," *CVPR*, 2019.

[128] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *CVPR*, 2017.

[129] P. P. Srinivasan, M. W. Tao, R. Ng, and R. Ramamoorthi, "Oriented light-field windows for scene flow," *ICCV*, 2015.

[130] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "Nerv: Neural reflectance and visibility fields for relighting and view synthesis," *arXiv*, 2020.

[131] P. P. Srinivasan, B. Mildenhall, M. Tancik, J. T. Barron, R. Tucker, and N. Snavely, "Lighthouse: Predicting lighting volumes for spatially-coherent illumination," *CVPR*, 2020.

[132] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," *CVPR*, 2019.

[133] P. P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, and R. Ng, "Learning to synthesize a 4D rgbd light field from a single image," *ICCV*, 2017.

[134] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, 2007.

[135] R. Swaminathan, S. B. Kang, R. Szeliski, A. Criminisi, and S. K. Nayar, "On the motion and appearance of specularities in image sequences," *ECCV*, 2002.

[136] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," *IJCV*, 1999.

[137] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3D models from single images wih a convolutional network," *ECCV*, 2016.

[138] T. Thonat, E. Shechtman, S. Paris, and G. Drettakis, "Multi-view inpainting for image-based scene editing and rendering," *3DV*, 2016.

[139] T. Totsuka and M. Levoy, "Frequency domain volume rendering," *Proceedings of SIGGRAPH*, 1993.

[140] R. Tucker and N. Snavely, "Single-view view synthesis with multiplane images," *CVPR*, 2020.

[141] S. Tulsiani, R. Tucker, and N. Snavely, "Layer-structured 3D scene inference via view synthesis," *ECCV*, 2018.

[142] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," *CVPR*, 2017.

[143] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

[144] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," *ECCV*, 2014.

[145] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," *CVPR*, 2018.

[146] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, 2004.

[147] Y. Weiss and W. T. Freeman, "What makes a good model of natural images?" *CVPR*, 2007.

[148] G. Wetzstein, D. Lanman, W. Heidrich, and R. Raskar, "Layered 3D: Tomographic image synthesis for attenuation-based light field and high dynamic range displays," *SIGGRAPH*, 2011.

[149] G. Wetzstein, D. Lanman, M. Hirsch, and R. Raskar, "Tensor displays: Compressive light field synthesis using multilayer displays with directional backlighting," *SIGGRAPH*, 2012.

[150] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Levoy, and M. Horowitz, "High performance imaging using large camera arrays," *SIGGRAPH*, 2005.

[151] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle, "Surface light fields for 3D photography," *SIGGRAPH*, 2000.

[152] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu, "Light field reconstruction using deep convolutional network on epi," *CVPR*, 2017.

[153] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," *NIPS*, 2016.

[154] J. Xie, R. Girshick, and A. Farhadi, "Deep3D: fully automatic 2D-to-3D video conversion with deep convolutional neural networks," *ECCV*, 2016.

[155] Z. Xu, S. Bi, K. Sunkavalli, S. Hadap, H. Su, and R. Ramamoorthi, "Deep view synthesis from sparse photometric images," *SIGGRAPH*, 2019.

[156] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: learning single-view 3D object reconstruction without 3D supervision," *NIPS*, 2016.

[157] B. Yang, Z. Lai, X. Lu, S. Lin, H. Wen, A. Markham, and N. Trigoni, "Learning 3d scene semantics and structure from a single depth image," *CVPR Workshops*, 2018.

[158] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee, "Weakly-supervised disentangling with recurrent transformations for 3D view synthesis," *NIPS*, 2015.

[159] H. W. F. Yeung, J. Hou, J. Chen, Y. Y. Chung, and X. Chen, "Fast light field reconstruction with deep coarse-to-fine modeling of spatial-angular clues," *ECCV*, 2018.

[160] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *ICLR*, 2016.

[161] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," *CVPR*, 2018.

[162] C. Zhang and T. Chen, "Spectral analysis for sampling image-based rendering data," *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

[163] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *CVPR*, 2018.

[164] Z. Zhang, Y. Liu, and Q. Dai, "Light field from micro-baseline image pair," *CVPR*, 2015.

[165] E. D. Zhong, T. Bepler, J. H. Davis, and B. Berger, "Reconstructing continuous distributions of 3D protein structure from cryo-EM images," *ICLR*, 2020.

[166] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Transactions on Graphics (SIGGRAPH)*, 2018.

[167] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," *ECCV*, 2016.

[168] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics (SIGGRAPH)*, 2004.

[169] M. Zwicker, W. Matusik, F. Durand, and H. Pfister, "Antialiasing for automultiscopic 3D displays," *Proceedings of Eurographics Symposium on Rendering*, 2006.